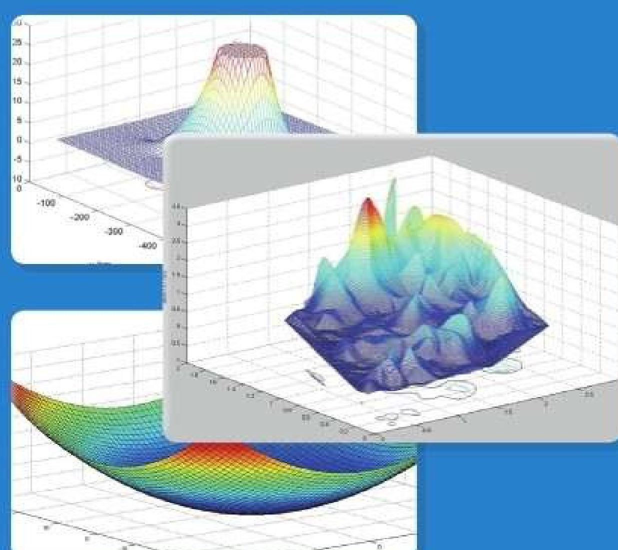
一 样 的 软 件， 不 一 样 的 学 习 方 法

精通 Matlab

数字图像处理与识别

张铮 倪红霞 苑春苗 杨立红 编著

- 内容涉及数字图像处理和识别技术的方方面面，包括图像的点运算、几何变换、空域和频域滤波、小波变换、图像复原、形态学处理、图像分割，以及图像特征提取的相关内容。
- 重点介绍了目前在工程技术领域非常流行的分类技术—— 人工神经网络（ANN）和支持向量机（SVM），并讲解了人脸识别技术。



CD-ROM 源程序

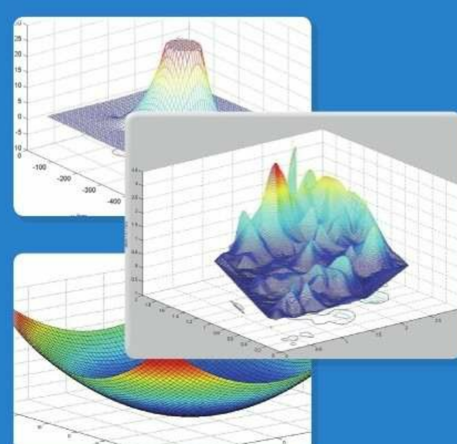
人民邮电出版社
POSTS & TELECOM PRESS

一 样 的 软 件， 不 一 样 的 学 习 方 法

精通 Matlab
数字图像处理与识别

张铮 倪红霞 苑春苗 杨立红 编著

- 内容涉及数字图像处理和识别技术的方方面面，包括图像的点运算、几何变换、空域和频域滤波、小波变换、图像复原、形态学处理、图像分割，以及图像特征提取的相关内容。
- 重点介绍了目前在工程技术领域非常流行的分类技术—— 人工神经网络（ANN）和支持向量机（SVM），并讲解了人脸识别技术。



CD-ROM 源程序

人民邮电出版社
POSTS & TELECOM PRESS

精通Matlab数字图像处理与识别

张铮　倪红霞　苑春苗　杨立红　编著

人民邮电出版社

北京

前言

图像处理与识别是当今计算机科学中一个热门研究方向，其应用广泛，发展前景乐观。Matlab是MathWorks公司开发的一款工程数学计算软件。由于其强大的科学运算、灵活的程序设计流程、高质量的图形可视化与界面设计，以及便捷的与其他程序和语言接口的功能，Matlab 已成为当今国际上科学界最具影响力、最有活力的软件。本书正是紧密结合大量的 Matlab 代码和案例展开对数字图像处理和识别技术的介绍。

让计算机理解所“看”到的东西是一件非常神秘和令人兴奋的事情。但要掌握好数字图像处理与识别技术却并非易事，它的理论性较强、门槛较高，在各个高校中，这门课程多是作为计算机专业研究生的选修课程。要理解该领域的知识，读者需要具有一定的数学基础，除此之外，还涉及信号处理、统计分析、模式识别和机器学习等专业领域知识，因此，令很多人望而却步。

其实“难以理解”的关键在于缺乏必要的先序知识，造成了读者在相关知识上难以跨越的鸿沟。我们在撰写本书过程中，对于可能造成读者理解困难的地方，均给出了必要的先序知识，尽量定性地去描述；对于那些并不一目了然的结论均给出了思路和解释；对于某些非常专业、已经超过本书讨论范围的相关知识也都在全书最后给出了参考文献，供有兴趣的读者进一步学习和研究。

本书的宗旨是，向读者介绍知识的同时，培养读者的思维方法，使读者知其然还要知其所以然，并在解决实际问题中能有自己的想法。

—— 内容安排

全书共分15章，主要内容介绍如下。

第1～2章介绍了数字图像处理的基础知识和Matlab数字图像编程基础，使读者第一步能够建立起对于数字图像本质的正确认识，了解和掌

握必要的术语，并且熟悉本书自始至终需要使用的工具Matlab，重点介绍其数字图像处理工具箱。

第3～4章分别介绍了图像的灰度变换和几何变换。通过灰度变换可以有效改善图像的外观，并在一定程度上实现图像的灰度归一化；几何变换则主要应用在图像的几何归一化和图像校准当中。总体而言，这些内容大多作为图像的前期预处理工作的一部分，是图像处理中相对固定和程式化的内容。

第5～6章分别从空间域和频率域两个角度去考量图像增强的各个主要方面。图像增强作为数字图像处理中相对简单却最具艺术性的领域之一，可理解为根据特定的需要突出一幅图像中的某些信息，同时削弱或去除某些不需要的信息的处理方法。其主要目的是，使处理后的图像对某种特定的应用来说，比原始图像更适用。

第7章小波变换继第6章之后继续在频率域中研究图像。傅立叶变换一直是频率域图像处理的基石，它能用正弦函数之和表示任何分析函数，而小波变换则基于一些有限宽度的基小波，这些小波不仅在频率上是变化的，而且具有有限的持续时间。例如，对于一张乐谱，小波变换不仅能提供要演奏的音符，而且说明了何时演奏等细节信息，但傅立叶变换只提供了音符，局部信息在变换中丢失。

第8章图像复原与图像增强相似，其目的也是改善图像质量。但是图像复原是试图利用退化过程的先验知识使已被退化的图像恢复本来面目，而图像增强是用某种试探的方式改善图像质量，以适应人眼的视觉与心理。引起图像退化的因素包括由光学系统、运动等造成的图像模糊，以及源自电路和光学因素的噪声等。图像复原是基于图像退化的数学模型，复原的方法也建立在比较严格的数学推导上。

第9章是本书中相对独立的一章，以介绍色彩模型之间的相互转换，以及彩色图像处理方面的基本概念和基本方法为主。随着基于互联网的图像处理应用在不断增长，彩色图像处理已经成为一个重要领域。

第10～12章（形态学处理、边缘检测与图像分割、特征提取）是从单纯图像处理向图像识别（机器视觉）的过渡，这一阶段的特点是，输入是图像，输出则是在识别意义上我们感兴趣的图像元素。形态学处理是提取图像元素的有力技术，它在表现和描述形状方面非常有用；分割过程则将一幅图像划分为组成部分或目标对象；研究特征提取则是要将前面提取出来的图像元素或目标对象表示为适合计算机后续处理的数值形式，最终形成能够直接供分类器使用的特征。

第13章在前面知识的基础之上，引出了机器视觉的前导性内容，给出了解决识别问题的一般思路。

第14～15章（人工神经网络、支持向量机）介绍了两种十分强大的分类技术，并介绍了人脸识别技术及经典案例应用。

—— 读者对象

● 具备必要的数学基础的相关专业的本科生、研究生。

● 工作在图像处理和识别领域一线的工程技术人员。

● 对于数字图像处理和机器视觉感兴趣的并且具备必要预备知识的所有读者。

—— 在阅读本书之前，读者最好具有如下的预备知识

● 读者应具备一定的数学基础，如高等数学知识、少量的线性代数基本概念加上对于概率理论主要思想的理解（识别部分）。

—— 在线支持和读者反馈

本书所有Matlab实例的源代码均可在bbs.book95.com网站的“金羽图书与答疑板块”板块与本书同名的主题帖子附件中提供下载。虽然本书中的所有例子都已经在Windows XP、Windows 2003和Windows 7等操作系统下的Matlab2006到MatlabR2011a的各个版本中测试通过，但由于交稿时间要求和笔者水平的局限，也有存在 Bug 的可能，即便正确也很可能存在更加优化的算法或更加合理的程序结构，如发现任何上述问题，请您不吝告知本书的作者（aaron@book95.com），以便我们做出改进。

—— 致谢

首先要感谢我的授业恩师——南开大学的白刚教授和天津大学的赵政教授，是他们引导我进入了图像处理与机器视觉的研究领域。同时，他们在我写作过程中的指点和教诲确保了本书内容的严谨。

本书的第7章、第8章由王艳艳和赵国宇编写，在此向他们表示感谢；同时感谢我的好友王艳平提供并调试了许多实例代码；感谢我的同事——闵卫东、陈香凝、任淑霞、姚清爽、王佳欣、孙连坤、孙学梅、张振和王作为等为本书所做出的工作；感谢我昔日的师弟和师妹——王杉、闫丽霞、刘旭、赵国宇和李宏鹏等参与了部分章节的编写和修改；感谢罗小科先生为本书制作了很多插图；感谢我的兄长张钊为本书提供了部分照片；还要感谢徐超、王欣和郭朋博士等为本书的编写提出了很多的宝贵意见和建议。

最后感谢我的妻子马宏、儿子张垚淼以及我的家人，没有你们的鼓励和支持就不会有我的这部作品。

编者

目　录

第1章 初识数字图像处理与识别

图像是指能在人的视觉系统中产生视觉印象的客观对象，包括自然景物、拍摄到的图片、用数学方法描述的图形等。图像的要素有几何要素（刻画对象的轮廓、形状等）和非几何要素（刻画对象的颜色、材质等）。

本章中，我们主要讲解数字图像和数字图像处理的实质内容和一般步骤，以及一些后面会经常使用到的基本概念。

1.1 数字图像

自然界中的图像都是模拟量，在计算机普遍应用之前，电视、电影、照相机等图像记录与传输设备都是使用模拟信号对图像进行处理。但是，计算机只能处理数字量，而不能直接处理模拟图像。所以我们要在使用计算机处理图像之前进行图像数字化

1.1.1 什么是数字图像

简单地说，数字图像就是能够在计算机上显示和处理的图像，可根据其特性分为两大类——位图和矢量图。位图通常使用数字阵列来表示，常见格式有**BMP**、**JPG**、**GIF**等；矢量图由矢量数据库表示，我们接触最多的就是**PNG**图形。

提示

本书只涉及数字图像中位图图像的处理与识别，如无特别说明，后文提到的“图像”和“数字图像”都仅仅是指位图图像。一般而言，使用数字摄像机或数字照相机得到的图像都是位图图像。

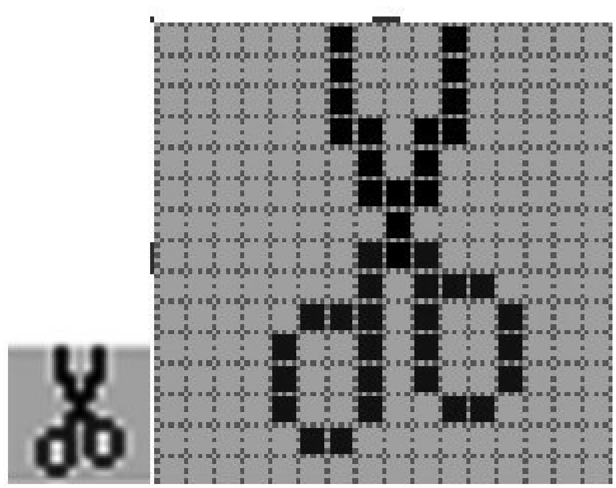
我们可以将一幅图像视为一个二维函数 f(x, y)，其中x和y是空间坐

标，而在x—y平面中的任意一对空间坐标(x, y)上的幅值f称为该点图像的灰度、亮度或强度。此时，如果f、x、y均为非负有限离散，则称该图像为数字图像（位图）。

一个大小为M×N数字图像是由M行N列的有限元素组成的，每个元素都有特定的位置和幅值，代表了其所在行列位置上的图像物理信息，如灰度和色彩等。这些元素称为图像元素或像素。

1.1.2 数字图像的显示

不论是CRT显示器还是LCD显示器，都是由许多点构成的，显示图像时这些点对应着图像的像素，称显示器为位映像设备。所谓位映像，就是一个二维的像素矩阵，而位图也就是采用位映像方法显示和存储的图像。当一幅数字图像被放大后就可以明显地看出图像是由很多方格形状的像素构成的，如图1.1所示。



▲图1.1 位图图像示例

1.1.3 数字图像的分类

根据每个像素所代表信息的不同，可将图像分为二值图像、灰度图

像、RGB 图像以及索引图像等。

1．二值图像

每个像素只有黑、白两种颜色的图像称为二值图像。在二值图像中，像素只有0和1两种取值，一般用0来表示黑色，用1表示白色。

2．灰度图像

在二值图像中进一步加入许多介于黑色与白色之间的颜色深度，就构成了灰度图像。这类图像通常显示为从最暗黑色到最亮的白色的灰度，每种灰度（颜色深度）称为一个灰度级，通常用L表示。在灰度图像中，像素可以取0～L-1之间的整数值，根据保存灰度数值所使用的数据类型不同，可能有256种取值或者说2^k种取值，当k=1时即退化为二值图像。

3．**RGB**图像

众所周知，自然界中几乎所有颜色都可以由红（Red, R）、绿（Green, G）、蓝（Blue, B）3种颜色组合而成，通常称它们为 RGB 三原色。计算机显示彩色图像时采用最多的就是 RGB 模型，对于每个像素，通过控制R、G、B三原色的合成比例决定该像素的最终显示颜色。

对于三原色RGB中的每一种颜色，可以像灰度图那样使用L个等级来表示含有这种颜色成分的多少。例如对于含有256个等级的红色，0表示不含红色成份，255表示含有100%的红色成分。同样，绿色和蓝色也可以划分为256个等级。这样每种原色可以用8位二进制数据表示，于是三原色总共需要 24位二进制数，这样能够表示出的颜色种类数目为$256\times256\times256=2^{24}$，大约有1 600万种，已经远远超过普通人所能分辨出的颜色数目。

RGB颜色代码可以使用十六进制数减少书写长度，按照两位一组的方式依次书写R、G、B三种颜色的级别。例如，0xFF0000代表纯红色，0x00FF00代表纯绿色，而0x00FFFF是青色（这是绿色和蓝色的加和）。当RGB三种颜色的浓度一致时，所表示的颜色就退化为灰度，例

如 0 x808080就是50%的灰色，0x000000为黑色，而0xFFFFFF为白色。常见颜色的RGB组合值如表1.1所示。

表1.1 常见颜色的RGB组合值

颜　色	R	G	B
红（0xFF0000）	255	0	0
蓝（0x00FF00）	0	255	0

续表

颜　色	R	G	B
绿（0x0000FF）	0	0	255
黄（0xFFFF00）	255	255	0
紫（0xFF00FF）	255	0	255
青（0x00FFFF）	0	255	255
白（0XFFFFFF）	255	255	255
黑（0x000000）	0	0	0
灰（0x808080）	128	128	128

未经压缩的原始BMP文件就是使用RGB标准给出的3个数值来存储图像数据的，称为**RGB**图像。在RGB图像中每个像素都是用24位二进制数表示，故也称为24位真彩色图像。

4．索引图像

如果对每个像素都直接使用24位二进制数表示，图像文件的体积将变得十分庞大。来看一个例子，对一个长、宽各为200像素，颜色数为16的彩色图像，每个像素都用RGB 3个分量表示。这样每个像素由3字节表示，整个图像就是 200×200×3=120KB。这种完全未经压缩的表示方式浪费了大量的存储空间。下面简单介绍另一种更节省空间的存储方式：索引图像。

同样还是对200×200像素的16色图像，由于这张图片中最多只有16

种颜色，那么可以用一张颜色表（16×3的二维数组）保存这16种颜色对应的RGB值，在表示图像的矩阵中使用那16种颜色在颜色表中的索引（偏移量）作为数据写入相应的行列位置。例如，颜色表中第3个元素为0xAA1111，那么在图像中所有颜色为0xAA1111的像素均可以由3-1=2表示（颜色表索引下标从0开始）。这样一来，每一个像素所需要使用的二进制数就仅仅为4位（0.5字节），从而整个图像只需要200×200×0.5=20KB就可以存储，而不会影响显示质量。

上文所指的颜色表就是常说的调色板（**Palette**），另一种说法叫做颜色查找表（**LUT：Look Up Table**）。Windows 位图中应用到了调色板技术。其实不仅是 Windows 位图，许多其他的图像文件格式如PCX、TIF、GIF都应用了这种技术。

在实际应用中，调色板中通常只有少于256种颜色。在使用许多图像编辑工具生成或者编辑GIF文件的时候，常常会提示用户选择文件包含的颜色数目。当选择较低的颜色数目时，将会有效地降低图像文件的体积，但也会一定程度上降低图像的质量。

使用调色板技术可以减小图像文件体积的条件是图像的像素数目相对较多，而颜色种类相对较少。如果一个图像中用到了全部的24位真彩色，对其使用颜色查找表技术是完全没有意义的，单纯从颜色角度对其进行压缩是不可能的。

1.1.4 数字图像的实质

实际上，1.1.1小节中对于数字图像f(x, y)的定义仅适用于最为一般的情况，即静态的灰度图像。更严格地说，数字图像可以是两个变量（对于静止图像，static image）或3个变量（对于动态画面，video sequence）的离散函数。在静态图像的情况下是f (x, y)，而如果是动态画面，则还需要时间参数t，即f (x, y, t)；函数值可能是一个数值（对于

灰度图像），也可能是一个向量（对于彩色图像）。

提示

静态的灰度图像是本书研究的主要对象，对于函数值为向量的情况会在第9章彩色图像处理中阐述。

图像处理是一个涉及诸多研究领域的交叉学科，下面就让我们从不同的角度来审视数字图像。

● 从线性代数和矩阵论的角度，数字图像是由图像信息组成的二维矩阵，矩阵的每个元素代表对应位置上的图像亮度和/或色彩信息。当然，这个二维矩阵在数据表示和存储上可能不是二维的，这是因为每个单位位置的图像信息可能需要不只一个数值来表示，这样可能需要一个三维矩阵来对其进行表示（参见2.2节关于Matlab中RGB图像表示的介绍）。

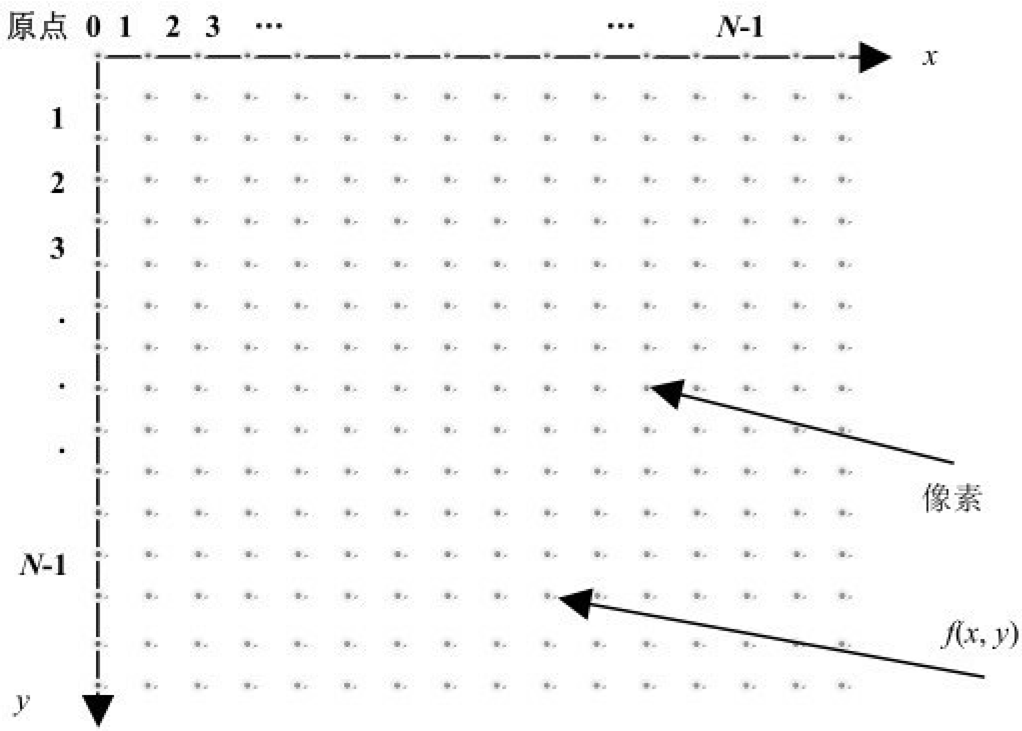
● 由于随机变化和噪声的原因，图像在本质上是统计性的。因而有时将图像函数作为随机过程的实现来观察存在其优越性。这时有关图像信息量和冗余的问题可以用概率分布和相关函数来描述和考虑。例如，如果知道概率分布，可以用熵（Entropy）H① 来度量图像的信息量，这是信息论中一个重要的思想。

【注】① 熵（Entropy）：熵是信息论中用于度量信息量的一个概念。一个系统越是有序，信息熵就越低；反之，一个系统越是混乱，信息熵就越高。所以，信息熵也可以说是系统有序化程度的一个度量。

● 从线性系统的角度考虑，图像及其处理也可以表示为用狄拉克冲激公式表达的点展开函数的叠加，在使用这种方式对图像进行表示时，可以采用成熟的线性系统理论研究。在大多数时候，我们都考虑使用线性系统近似的方式对图像进行近似处理以简化算法。虽然实际的图像并不是线性的，图像坐标和图像函数的取值都是有限的和非连续的。

1.1.5 数字图像的表示

为了表述像素之间的相对和绝对位置，我们通常还需要对像素的位置进行坐标约定。本书中所使用的坐标约定如图1.2所示。但在Matlab中坐标的约定会有变化，具体请参见2.1.5小节。



▲图1.2 数字图像的坐标约定

在这之后，一幅物理图像就被转化成了数字矩阵，从而成为计算机能够处理的对象了。数字图像f的矩阵的表示如下。

$$f(y,x)=\begin{bmatrix} f(0,0) & \cdots & f(0,N-1) \\ \vdots & \ddots & \vdots \\ f(M-1,0) & \cdots & f(M-1,N-1) \end{bmatrix} \qquad (1\text{-}1)$$

有时也可以使用传统矩阵表示法来表示数字图像和像素。

$$A=\begin{bmatrix} a_{0,0} & \cdots & a_{0,N-1} \\ \vdots & \ddots & \vdots \\ a_{M-1,0} & \cdots & a_{M-1,N-1} \end{bmatrix} \tag{1-2}$$

其中行列（M行N列）必须为正整数，而离散灰度级数目L一般为2的k次幂，k为整数（因为使用二进制整数值表示灰度值），图像的动态范围为[0,L-1]，那么图像存储所需的比特数为b=M×N×k。注意到在矩阵f（y, x）中，一般习惯于先行下标，后列下标的表示方法，因此这里先是纵坐标y（对应行），然后才是横坐标x（对应列）。

而有些图像矩阵中，很多像素的值都是相同的。例如一个纯黑背景上使用不同灰度勾勒的图像，大多数像素的值都会是 0。这种矩阵称为稀疏矩阵（Sparse Matrix），可以通过简单描述非零元素的值和位置来代替大量地写入0元素。这时存储图像需要的比特数可能大大减少。

1.1.6 图像的空间和灰度级分辨率

1．图像的空间分辨率

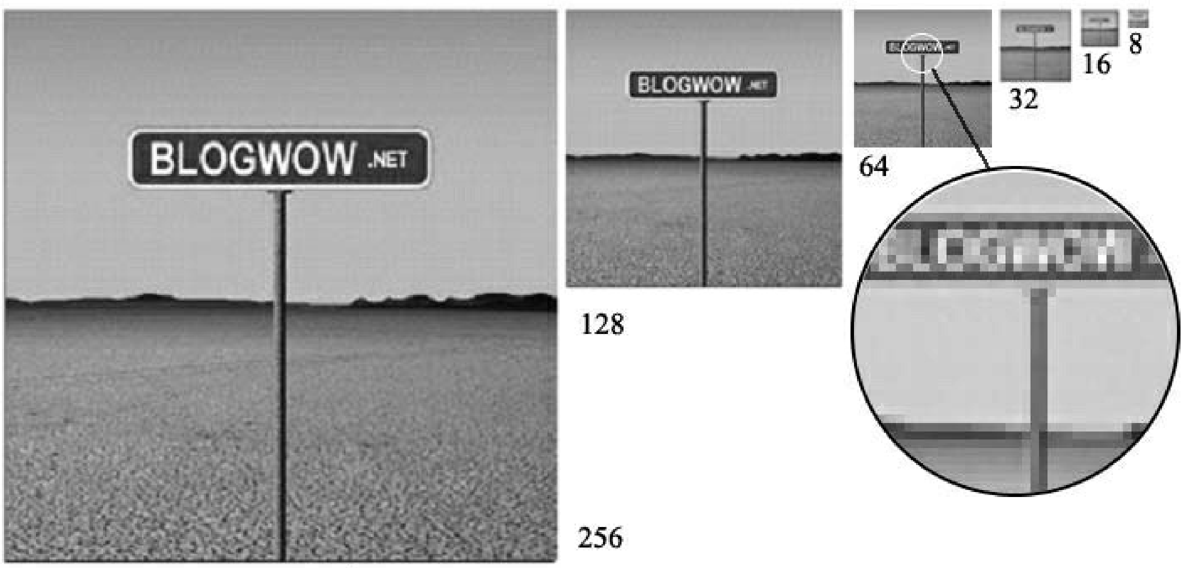
图像的空间分辨率（**Spatial Resolution**）是指图像中每单位长度所包含的像素或点的数目，常以像素/英寸（pixels per inch, ppi）为单位来表示。如 72ppi表示图像中每英寸包含72个像素或点。分辨率越高，图像将越清晰，图像文件所需的磁盘空间也越大，编辑和处理所需的时间也越长。

像素越小，单位长度所包含的像素数据就越多，分辨率也就越高，但同样物理大小范围内所对应图像的尺寸也会越大，存储图像所需要的字节数也越多。因而，在图像的放大缩小算法中，放大就是对图像的过采样，缩小是对图像的欠采样，这些会在4.5节图像缩放中进一步介绍。

一般在没有必要对涉及像素的物理分辨率进行实际度量时，通常会

称一幅大小为M×N的数字图像的空间分辨率为M×N像素。

图1.3所示为同一幅图像在不同的空间分辨率下呈现出的不同效果。当高分辨率下的图像以低分辨率表示时，在同等的显示或者打印输出条件下，图像的尺寸变小，细节变得不明显；而当将低分辨率下的图像放大时，则会导致图像的细节仍然模糊，只是尺寸变大。这是因为缩小的图像已经丢失了大量的信息，在放大图像时只能通过复制行列的插值方法来确定新增像素的取值。



▲图1.3 图像的空间分辨率（一幅分辨率为 256×256的图像逐次减少至8×8的分辨率）

2．图像的灰度级/辐射计量分辨率

在数字图像处理中，灰度级分辨率又叫色阶，是指图像中可分辨的灰度级数目，即前文提到的灰度级数目L，它与存储灰度级别所使用的数据类型有关。由于灰度级度量的是投射到传感器上光辐射值的强度，所以灰度级分辨率也叫辐射计量分辨率（**Radiometric Resolution**）。

随着图像的灰度级分辨率逐渐降低，图像中包含的颜色数目变少，从而在颜色的角度造成图像信息受损，同样使图像细节表达受到了一定

影响，如图1.4所示。



▲图1.4 图像的灰度级分辨率（分别具有 256、32、16、8、4和 2个灰度级的一幅图像）

1.2 数字图像处理与识别

1.2.1 从图像处理到图像识别

图像处理、图像分析和图像识别是认知科学与计算机科学中的一个令人兴奋的活跃分支。从1970年这个领域经历了人们对其兴趣的爆炸性增长以来，到20世纪末逐渐步入成熟。其中，遥感、技术诊断、智能车自主导航、医学平面和立体成像以及自动监视领域是发展最快的一些方向。这种进展最集中地体现在市场上多种应用这类技术的产品的纷纷涌

现。事实上，从数字图像处理到数字图像分析，再发展到最前沿的图像识别技术，其核心都是对数字图像中所含有的信息的提取及与其相关的各种辅助过程。

1．数字图像处理

数字图像处理（**Digital Image Processing**）就是指使用电子计算机对量化的数字图像进行处理，具体地说就是通过对图像进行各种加工来改善图像的外观，是对图像的修改和增强。

图像处理的输入是从传感器或其他来源获取的原始的数字图像，输出是经过处理后的输出图像。处理的目的可能是使输出图像具有更好的效果，以便于人的观察；也可能是为图像分析和识别做准备，此时的图像处理是作为一种预处理步骤，输出图像将进一步供其他图像分析、识别算法使用。

2．数字图像分析

数字图像分析（**Digital Image Analyzing**）是指对图像中感兴趣的目标进行检测和测量，以获得客观的信息。数字图像分析通常是指将一幅图像转化为另一种非图像的抽象形式，如图像中某物体与测量者的距离以及目标对象的计数或其尺寸等。这一概念的外延包括边缘检测和图像分割、特征提取以及几何测量与计数等。

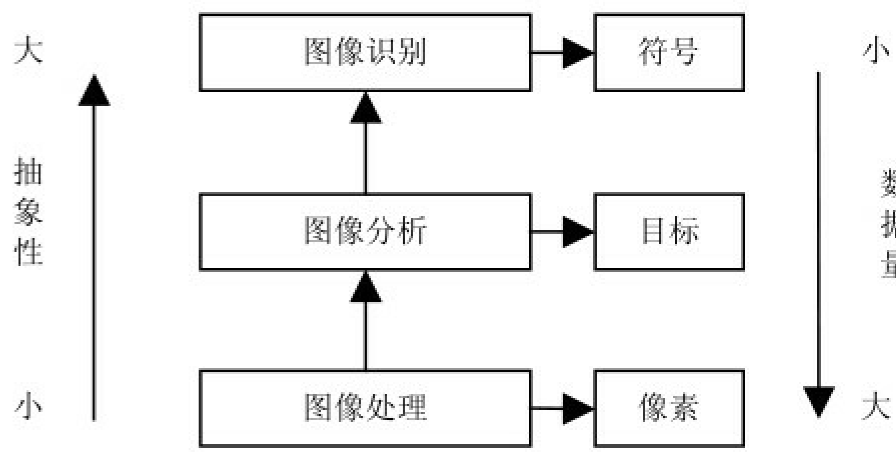
图像分析的输入是经过处理的数字图像，其输出通常不再是数字图像，而是一系列与目标相关的图像特征（目标的描述），如目标的长度、颜色、曲率和个数等。

3．数字图像识别

数字图像识别（**Digital Image Recognition**）主要研究图像中各目标的性质和相互关系，识别出目标对象的类别，从而理解图像的含义。这往往囊括了使用数字图像处理技术的很多应用项目，如光学字符识别（OCR）、产品质量检验、人脸识别、自动驾驶、医学图像和地貌图像的自动判读理解等。

图像识别是图像分析的延伸，它根据从图像分析中得到的相关描述（特征）对目标进行归类，输出我们感兴趣的目标类别标号信息（符号）。

总而言之，从图像处理到图像分析再到图像识别这个过程，是一个将所含信息抽象化，尝试降低信息熵，提炼有效数据的过程，如图1.5所示。



▲图1.5 数字图像处理、分析和识别的关系

从信息论的角度上说，图像应当是物体所含信息的一个概括，而数字图像处理侧重于将这些概括的信息进行变换，例如升高或降低熵值。数字图像分析则是将这些信息抽取出来以供其他过程调用。当然，在不太严格时，数字图像处理也可以兼指图像处理和分析。

您或许也听过另一个概念，计算机图形学（**Computer Graphics**）。此概念与数字图像分析大致相反，它是一个对由概念或数学表述的物体图像进行处理和显示的过程。

1.2.2 数字图像处理与识别的应用实例

如今，数字图像处理与机器视觉的应用越来越广泛，已经渗透到国

家安全、航空航天、工业控制、医疗保健等各个领域乃至我们的日常生活和娱乐当中，在国民经济中发挥着举足轻重的作用。它的一些典型的应用如表1.2所示。

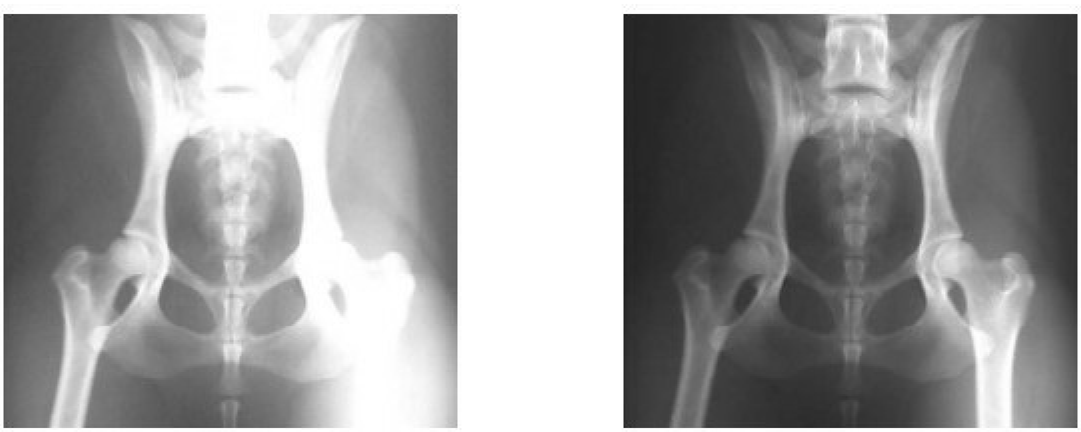
表1.2 图像处理与识别的典型应用

相关领域	典型应用
安全监控	指纹验证，基于人脸识别的门禁系统
工业控制	产品无损检测，商品自动分类
医疗保健	X 光照片增强，CT，核磁共振，病灶自动检测
生活娱乐	基于表情识别的笑脸自动检测，汽车自动驾驶，手写字符识别

下面结合2个典型的应用来说明。

1．图像处理的典型案例**——X**光照片的增强

图1.6所示为两幅图片，（a）是一幅直接拍摄未经处理的X光照片，对比度较低，图像细节难以辨识；（b）中呈现了（a）经过简单的增强处理后的效果，图像较为清晰，可以有效地指导诊断和治疗。从中读者应该可以看出图像处理技术在辅助医学成像上的重要作用。



（a） 未经处理的 X 光照片　　（b） 经过图像增强的 X 光照片

▲图1.6 图像处理前后的效果对比

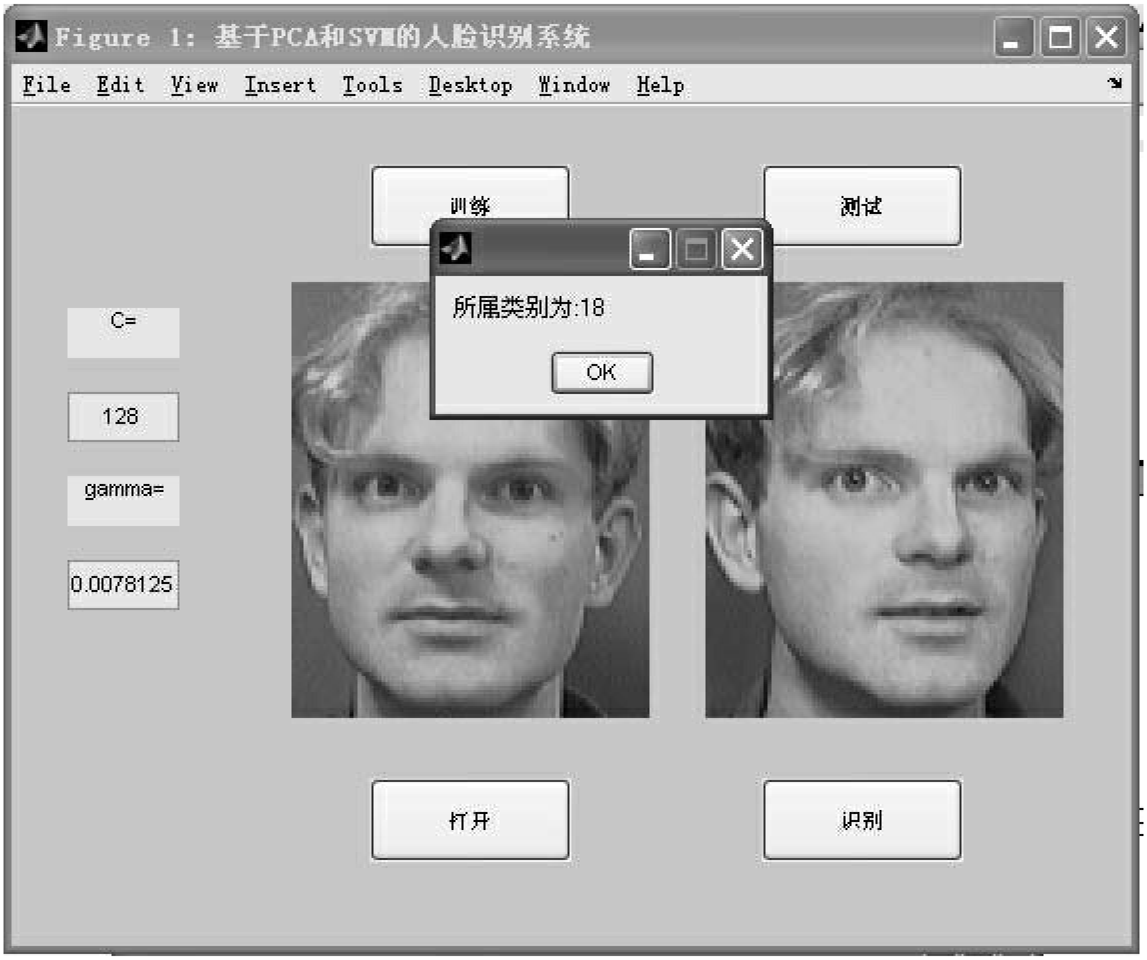
2．图像识别的典型案例**——**人脸识别

人脸识别技术就是以计算机为辅助手段，从静态图像或动态图像中识别人脸。问题一般可以描述为给定一个场景的静态或视频图像，利用已经存储的人脸数据库确认场景中的一个或多个人。一般来说，人脸识别研究分为3个部分：从具有复杂背景的场景中检测并分离出人脸所在的区域；抽取人脸识别特征；然后进行匹配和识别。

虽然人类从复杂背景中识别出人脸及表情相当容易，但人脸的自动机器识别却是一个极具挑战性的课题。它跨越了模式识别、图像处理、计算机视觉以及神经生理学、心理学等诸多研究领域。

如同人的指纹一样，人脸也具有唯一性，可用来鉴别一个人的身份，人脸识别技术在商业、法律和其他领域有着广泛的应用。目前，人脸识别已成为法律部门打击犯罪的有力工具，在毒品跟踪、反恐怖活动等监控中有着很大的应用价值。此外，人脸识别的商业应用价值也正在日益增长，主要是信用卡或者自动取款机的个人身份核对。与利用指纹、手掌、视网膜、虹膜等其他人体生物特征进行个人身份鉴别的方法相比，人脸识别具有直接、友好、方便的特点，特别是对于个人来说没有任何心理障碍。

图 1.7所示为本书后面综合案例中实现的一个基于主成分分析（Principal Component Analysis, PCA）和支持向量机（Support Vector Machine, SVM）的人脸识别系统的简单界面。



▲图1.7 一个简单的人脸识别系统

1.2.3 数字图像处理与识别的基本步骤

总体来说，数字图像处理与识别包括以下几项内容。

图像的点运算通过灰度变换可以有效改善图像的外观，并在一定程度上实现图像的灰度归一化。本书第3章中将介绍多种基于图像点运算的处理方法，如图像拉伸、对比度增强、直方图均衡以及直方图匹配等。

图像的几何变换主要应用在图像的几何归一化和图像校准当中，将在本书第4章中进行介绍。总体而言，第3、4两章内容大多作为图像的前期预处理工作必要的一部分，是图像处理中相对固定和程式化的内

容。

图像增强作为数字图像处理中相对简单却最具艺术性的领域之一，可理解为根据特定的需要突出一幅图像中的某些信息，同时削弱或去除某些不需要的信息的处理方法。其主要目的是使处理后的图像对某种特定的应用来说，比原始图像更适用。作为图像处理中一个相当主观的领域（增强的目的是可以让人更好地观察和认知图像），图像增强是以下多种图像处理方法的前提与基础，也是在图像获取后的先期步骤。本书第5～6章分别从空间域和频率域两个角度去考量图像增强的各个主要方面。

小波变换伴随着人们对图像压缩、边缘和特征检测以及纹理分析的需求的提高应运而生。傅立叶变换一直是频率域图像处理的基石，它能用正弦函数之和表示任何分析函数，而小波变换则基于一些有限宽度的基小波，这些小波不仅在频率上是变化的，而且具有有限的持续时间。例如对于一张乐谱，小波变换不仅能提供要演奏的音符，而且说明了何时演奏等细节信息，但是傅立叶变换只提供了音符，局部信息在变换中丢失。本书将在第7章中讨论小波变换。

图像复原与图像增强相似，其目的也是改善图像质量。但是图像复原是试图利用退化过程的先验知识使已被退化的图像恢复本来面目，而图像增强是用某种试探的方式改善图像质量，以适应人眼的视觉与心理。引起图像退化的因素包括由光学系统、运动等造成的图像模糊，以及源自电路和光学因素的噪声等。图像复原是基于图像退化的数学模型，复原的方法也建立在比较严格的数学推导上。本书第8章介绍图像复原。

彩色图像处理实际上是从图像的类型分类，主要包括对全彩图像的处理，也包括灰度图像的伪彩色化。彩色图像处理相对二值图像和灰度图像更为复杂，我们会在第9章中简要阐述这方面的基础知识。

形态学图像处理是一种将数学形态学推广应用于图像处理领域的新

方法，是一种基于物体自然形态的图像处理分析方法。而形态学的概念最早来源于生物学，是一门生物学中研究动物和植物结构的一个分支科学。数学形态学（也称图像代数）则是一种以形态为基础对图像进行分析的数学工具，其基本思想是用具有一定形态的结构元素去度量和提取图像中的对应形状以达到对图像分析和识别的目的。图像形态学往往用于边界提取、区域填充、连通分量的提取、凸壳、细化、像素化等图像操作。本书第10章专门介绍形态学图像处理的方法与基本应用，这一章将是我们从图像处理到图像特征处理的开始。

图像分割（**Image Division**）是指将一幅图像分解为若干互不交叠区域的过程，分割出的区域需要同时满足均匀性和连通性的条件。目标的表示与描述是指用组成目标区域的像素或区域边界的像素标出这一目标，并且对目标进行抽象描述，使计算机能充分利用所获得的处理分割结果。实际上，表达和描述的联系是十分紧密的，表达的方法限制了描述的精确性，而只有通过对目标的描述，各种表达方法才有意义。

特征提取（**Feature Extraction**）指的是进一步处理之前得到的图像区域和边缘，使其成为一种更适合于计算机处理的形式。为了使计算机能够“理解”图像，从而具有真正意义上的“视觉”，我们研究如何从图像中提取有用的数据或信息，得到图像的“非图像”的表示或描述，如数值、向量和符号等，这一过程就是特征提取，而提取出来的这些“非图像”的表示或描述就是特征。有了这些数值或向量形式特征我们就可以通过训练过程教会计算机如何懂得这些特征，从而使计算机具有了识别图像的本领。常用的图像特征有纹理特征、形状特征、空间关系特征等。

对象识别（**Object Recognition & Identification**）一般是指利用前一步从数字图像中提取出的特征向量进行分类和理解的过程，这涉及计算机技术、模式识别、人工智能等多方面的知识。这一步骤是建立在前面诸多步骤的基础上的，用以向上层控制算法提供最终所需的数据或直

接报告识别结果。事实上，对象识别已经上升到了机器视觉的层面上。在笔者曾参与的多个项目中，对象识别都被作为替代传统图像处理手段的方式，应用在人脸识别、表情识别等需求中。

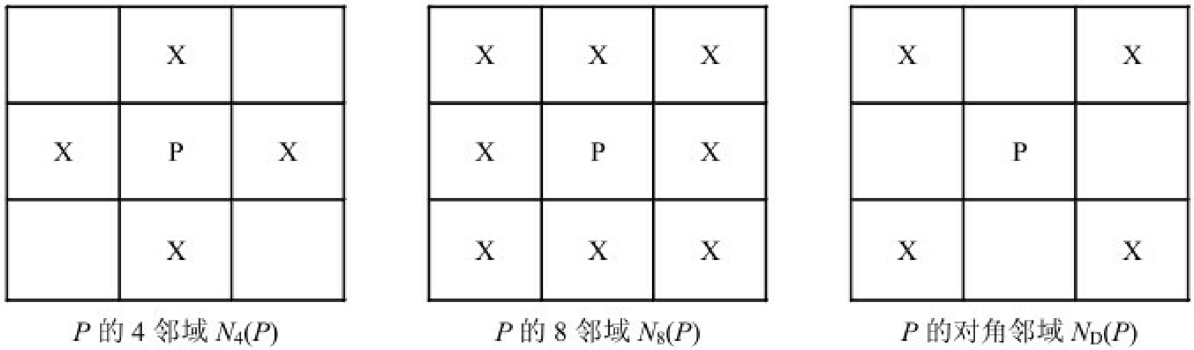
经过上述处理步骤，最初的一幅原始的、可能存在干扰和缺损的图像就变成了其他控制算法需要的信息，从而实现了图像理解的最终目的。以上概括了数字图像处理的基本顺序，但不是每个图像处理系统都一定需要进行所有这些步骤。事实上，很多图像处理系统并不需要处理彩色图像，或者不需要进行图像复原。在实际的图像处理系统设计中，我们应当根据实际需要决定采用哪些步骤和模块。

1.3 数字图像处理的预备知识

数字图像是由一组具有一定的空间位置关系的像素组成的，因而具有一些度量和拓扑性质。理解像素间的关系是学习图像处理的必要准备，这主要包括相邻像素、邻接性、连通性、区域、边界的概念，以及今后要用到的一些常见距离度量方法。另外1.3.3小节还将简单介绍几种基本的图像操作。

1.3.1 邻接性、连通性、区域和边界

为理解这些概念，需要首先了解相邻像素的概念。依据标准的不同，我们可以关注像素P的4邻域和8邻域，如图1.8所示。



▲图1.8 P的各种邻域

1．邻接性

定义V是用于决定邻接性（**Adjacency**）的灰度值集合，它是一种相似性的度量，用于确定所需判断邻接性的像素之间的相似程度。例如在二值图像中，如果我们认为只有灰度值为1的像素是相似的，则即 V={1}，当然相似性的规定具有主观标准，因此也可以认为 V={0，1}，此时邻接性完全由位置决定；而对于灰度图像，这个集合中则很可能包含更多的元素。此外，定义对角邻域$n_D(P)$为8-邻域中不属于4-邻域的部分（见图1.8），那么：

● 4邻接（4-Neighbor）：如果 $Q\in n_4(P)$，则称具有V中数值的两个像素P和Q是4邻接的。

● 8邻接（8-Neighbor）：如果 $Q\in n_8(P)$，则称具有V中数值的两个像素P和Q是8邻接的。

举例来说，图1.9（a）、（b）分别是像素和Q、Q_1、Q_2的4邻接和8邻接示意图。而对于两个图像子集S_1和S_2，如果S_1中的某些像素和S_2中的某些像素相邻，则称这两个子集是邻接的。



（a）4 邻接示意图　　　　（b）8 邻接示意图

▲图1.9

2．连通性

为了定义像素的连通性，我们首先需要定义像素P到像素Q的通路（Path）。这也是建立在邻接性的基础上的。

像素**P**到像素**Q**的通路（**Path**）指的是一个特定的像素序列(x_0, y_0), (x_1, y_1), …, (x_n, y_n)，其中$(x_0,y_0)=(x_p,y_p)$，$(x_n,y_n)=(x_q,y_q)$。并且像素(x_i,y_i)和(x_{i-1},y_{i-1})在满足$1≤i≤n$时是邻接的。在上面的定义中，n是通路的长度，若$(x_0, y_0) = (x_n, y_n)$，则这条通路是闭合通路。相对应于邻接的概念，在这里有4通路和8通路。这个定义和图论中的通路定义是基本相同的，只是由于邻接概念的加入而变得更加复杂。

像素的连通性（**Contiguous**）**：**令S代表一幅图像中的像素子集。如果在S中全部像素之间存在一个通路，则可以称2个像素P和Q在S中是连通的。此外，对于S中的任何像素P，S中连通到该像素的像素集叫做S的连通分量。如果S中仅有一个连通分量，则集合S叫做连通集。

3．区域和边界

区域的定义建立在连通集的基础上的。令R是图像中的一个像素子集，如果R同时是连通集，则称R为一个区域（**Region**）。

边界（**Boundary**）的概念是相对于区域而言的。一个区域的边界（或边缘、轮廓）是区域中所有有一个或多个不在区域R中的邻接像素的像素所组成的集合。显然，如果区域R是整幅图像，那么边界就由图

像的首行、首列、末行和末列定义。因此，通常情况下，区域指一幅图像的子集，并包括区域的边缘。而区域的边缘（**Edge**）由具有某些导数值的像素组成，是一个像素及其直接邻域的局部性质，是一个有大小和方向属性的矢量。

边界和边缘是不同的。边界是和区域有关的全局概念，而边缘表示图像函数的局部性质。

1.3.2 距离度量的几种方法

基于上一小节提到的相关知识，我们来理解距离度量的概念。假设对于像素$P(x_p, y_p)$、$Q(x_q, y_q)$、$R(x_r, y_r)$而言，有函数D满足如下3个条件，则函数D可被称为距离函数或度量。

（i）$D(P, Q)\geq 0$，当且仅当P=Q时有$D(P, Q) = 0$

（ii）$D(P, Q) = D(Q, P)$

（iii）$D(P, Q) \leq D(P, R) + D(R, Q)$

常见的几种距离函数如下。

（i）欧氏距离

$$D_e(P,O) = \sqrt{(x_p - x_q)^2 + (y_p - y_q)^2} \qquad (1\text{-}3)$$

即距离等于r的像素形成以P为圆心的圆。

（ii）D_4距离（街区距离）

$$D_4(P,Q) = |x_p - x_q| + |y_p - y_q| \qquad (1\text{-}4)$$

即距离等于r的像素形成以P为中心的菱形。

（iii）D_8距离（棋盘距离）

$$D_8(P,Q) = \max\left(\left|x_p - x_q\right| + \left|y_p - y_q\right|\right) \tag{1-5}$$

即距离等于r的像素形成以P为中心的方形。

距离度量参数可以用于对图像特征进行比较和分类或者进行某些像素级操作。最常用的距离度量是欧氏距离。然而在形态学中，也可能使用街区距离和棋盘距离。

1.3.3 基本的图像操作

在后续章节中，将涉及各种各样的图像操作，这里就几种最为典型和常用的图像操作着重说明。按照处理图像的数量分类，可以分为对单幅图像操作（如滤波）和对多幅图像操作（如求和、求差和逻辑运算等）；按照参与操作的像素范围的不同，可以分为点运算和邻域运算；而根据操作的数学性质，又可以分为线性操作和非线性操作。

1．点运算和邻域运算

点运算指的是对图像中的每一个像素逐个进行同样的灰度变换运算。设r和s分别是输入图像f(x, y)和输出图像g(x, y)在任一点(x, y)的灰度值，则点运算可以使用下式定义。

$$s = T(r) \tag{1-6}$$

而如果将点运算扩展，对图像中每一个小范围（邻域）内的像素进行灰度变换运算，即称为邻域运算或邻域滤波。这可以使用下式定义。

$$g(x, y) = T\,[f(x, y)] \tag{1-7}$$

我们将分别在第3章和第5章介绍点运算和邻域运算。

2．线性和非线性操作

令H是一种算子，其输入输出都是图像。若对于任意两幅（或两

组）图像f_1和f_2及任意两个标量a和b都有如下关系成立。

$$H(aF_1 + bF_2) = aH(F_1) + bH(F_2) \quad (1\text{-}8)$$

则称 H 为线性算子。也即对两幅图像的线性组合应用该算子与分别应用该算子后的图像在进行同样的线性组合所得到的结果相同，即算子 H 满足线性性质。同样的，不符合上述定义的算子即为非线性算子，对应的是非线性图像操作。举例来说，滤波中的平均平滑、高斯平滑、梯度锐化等都是线性运算，而中值滤波（详见第5章空间域图像增强）则是非线性的。

线性操作由于其稳定性的特点而在图像处理中占有非常重要的地位。尽管非线性算子常常也能够提供较好的性能，但它的不可预测性使其在一些如军事图像处理和医学图像处理等严格的应用领域中难以获得广泛的应用。

第2章 Matlab数字图像处理基础

Matlab是MathWorks公司开发的一款工程数学计算软件。不同于C++、Java、Fortran等高级编程语言是对机器行为进行描述，Matlab 是对数学操作进行更直接的描述。Matlab 图像处理工具箱（Image Processing Toolbox, IPT）封装了一系列针对不同图像处理需求的标准算法，它们都是通过直接或间接地调用Matlab中的矩阵运算和数值运算函数来完成图像处理任务的。

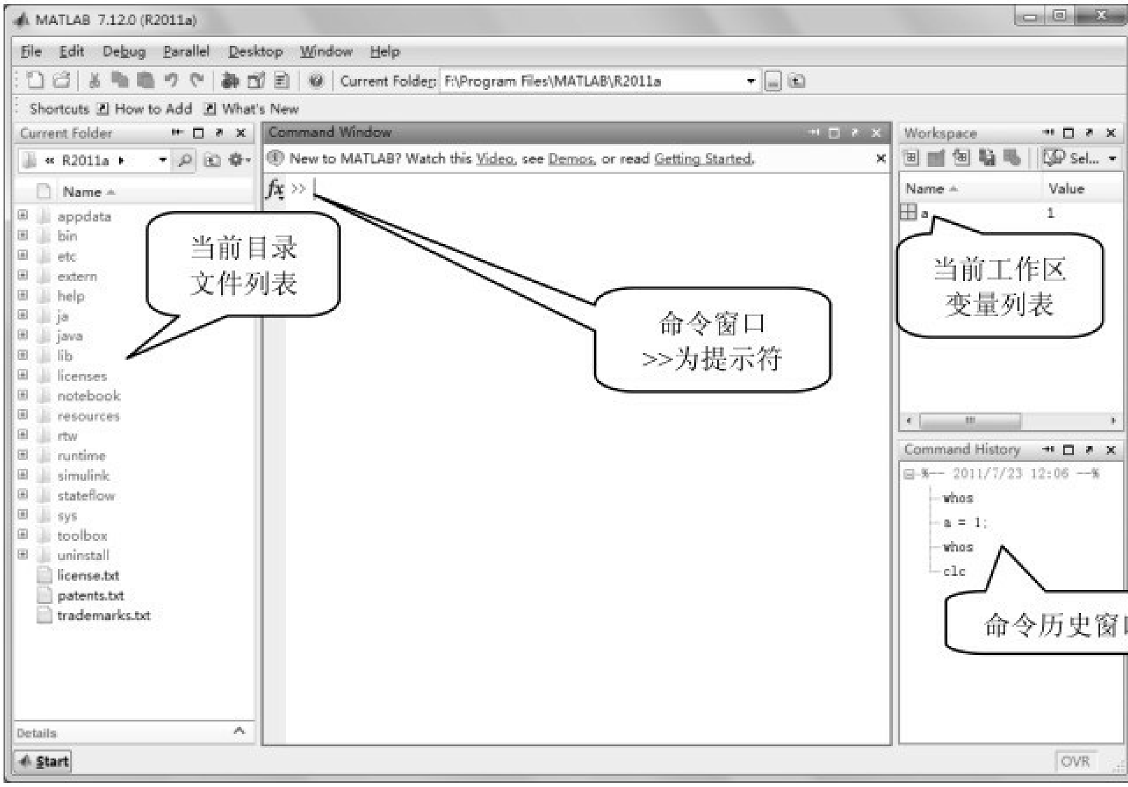
2.1 Matlab R2011a简介

本节将介绍一些Matlab R2011a中与图像处理密切相关的数据结构及基本操作，如基本文件操作、变量使用、程序流程控制、打开和关闭图像以及图像格式转换和存储方式等。这些都是后续将要学习的图像处理算法的基础。

2.1.1 Matlab软件环境

1．软件界面

图 2.1所示是运行于32-bit Windows操作系统上的Matlab R2011a界面。软件主界面由3个子窗口组成，左上为当前工作目录的文件列表，右上方为当前工作区的变量，右下为当前和最近会话的命令历史记录，而中间的主窗口则是命令输入和结果输出区，>>为提示符。



▲图2.1 Matlab界面

2．Matlab命令与程序

可以在>>提示符后面输入简单的算式（如5*3-2）或带有函数的算式（如sin(pi/2)*sqrt(3)/2）并按回车键，会提示ans=0.8660，这就是Matlab最基本的计算功能。

这样的输入形式实际上是**Matlab**命令，而如果在每行命令的结尾输入半角分号，命令窗口不会立即显示命令执行的结果，而会将结果保存在工作区中。示例如下。

>> res = sin(pi/2)*sqrt(3)/2;　% 将计算结果保存至变量res当中

此时，变量res已经存在于工作区中，但是命令窗口不会回显它的值。

另外，也可以在文件菜单下执行**“New”** → **“M-Files”**命令来创建一个新的Matlab文件，在里面输入命令（以半角分号结尾），从而得到一

个**Matlab**程序。在Matlab程序中，使用%表示注释，其用法和C/C++中的//注释符类似。

3．跨行语句

Matlab 允许在同一行中输入多条语句，之间用分号隔开。同时，Matlab 还允许将同一条语句分割在多行中书写以方便较长语句的阅读，方法是在行末使用3个半角圆点。示例如下。

```
>> z = 2 .* x +exp( x .^ 2 + y .^ 2 - sqrt(1 - log(x) - log (y) ) )...
        - y .* sqrt(t) - x .* sqrt(t);
```

2.1.2 文件操作

默认情况下，Matlab可以自动搜索到当前目录（Current Directory）和Matlab的路径变量 path中所含有目录下面的文件。对处在这些位置可由 Matlab 执行的文件，直接在命令窗口中键入文件名即可运行。如果需要直接运行其他目录下的文件，就要使用addpath和genpath等命令向路径列表中添加路径。

1．**addpath**函数

addpath函数向path变量中加入指定的目录路径，其原型如下。

addpath('dir','dir2','dir3' ...'-flag')

该函数可以接受任意数目的参数。

参数说明

● dir、dir2、dir3等为要加入的目录路径，这些变量必须是绝对路径。

● flag参数可以用来指定函数的行为，它是可选参数，其取值的含义如表2.1所示。

表2.1 addpath函数flag参数的取值

合法取值	含　义
0 或者 begin	这些路径将被添加到搜索列表的最前面，这些目录中含有的文件将先于原列表中的同名文件被找到从而执行。这往往用于需要修改系统某一命令行为的场合
1 或者 end	这些路径将被添加到搜索列表的最后面，原列表中的同名文件将先于这些目录中含有的文件被找到从而执行。这样可以避免用户 M 文件覆盖系统 M 文件的功能
省略	与 0 或 begin 相同

可以在使用addpath函数前后查看path变量的内容，以确定添加成功。

2．genpath函数

genpath函数生成包含指定目录下所有子目录的路径变量，其原型如下。

p = genpath('directory');

参数说明

● 参数directory为指定的目录。

返回值

● 函数返回包含指定目录本身和其全部子目录的数据。返回值也可以直接提供给addpath，从而直接添加一个目录及其全部子目录到当前路径列表中。通过这样的方式可以方便地调用我们自己的程序工具箱，例如使用下面的命令将目录“F:\doctor research\Matlab Work\FaceRec”添加到系统当前路径列表后，就可以直接调用人脸识别工具箱FaceRec中的任何函数了。

>> addpath(genpath('F:\doctor research\Matlab Work\FaceRec')) %注意这里要使用绝对路径

也可以在运行M文件时使用完整的文件路径，从而避免同名文件的冲突问题，或是从资源管理器中将M文件拖动到Matlab的命令窗口中直接运行。

3．打开与编辑**M**文件

如果需要编辑某个M文件，可以使用open命令和edit命令，它们的

调用形式如下。

open filename

edit filename

参数filename为需要打开的文件名。edit命令只能编辑M文件，而open命令可以使用Windows默认操作打开一系列其他类型的文件。

2.1.3 在线帮助的使用

在Matlab中，有4种方法获取软件的在线帮助。

1．help命令

help命令可以用于查看Matlab系统或M文件中内置的在线帮助信息。命令格式如下。

help command-name

command-name为需要查看在线帮助的命令或函数的名称。例如，想要查看doc命令的使用方法，可在命令提示符下直接输入 help doc，如图 2.2所示。

2．doc命令

doc 命令可以用于查看命令或函数的 HTML 帮助，这种帮助信息可以在帮助浏览器窗口中打开。其调用格式如下。

doc function-name

doc命令可提供比help命令更多的信息，还可能包含图片或视频等的多媒体例子，对图像处理工具箱中的函数更是如此。

图 2.3所示为在命令行中输入 doc imhist命令后出现的帮助示例界面。

```
>> help doc
  DOC Reference page in Help browser.

    DOC opens the Help browser, if it is not already running, and
    otherwise brings the Help browser to the top.

    DOC FUNCTIONNAME displays the reference page for FUNCTIONNAME in
    the Help browser. FUNCTIONNAME can be a function or block in an
    installed MathWorks product.

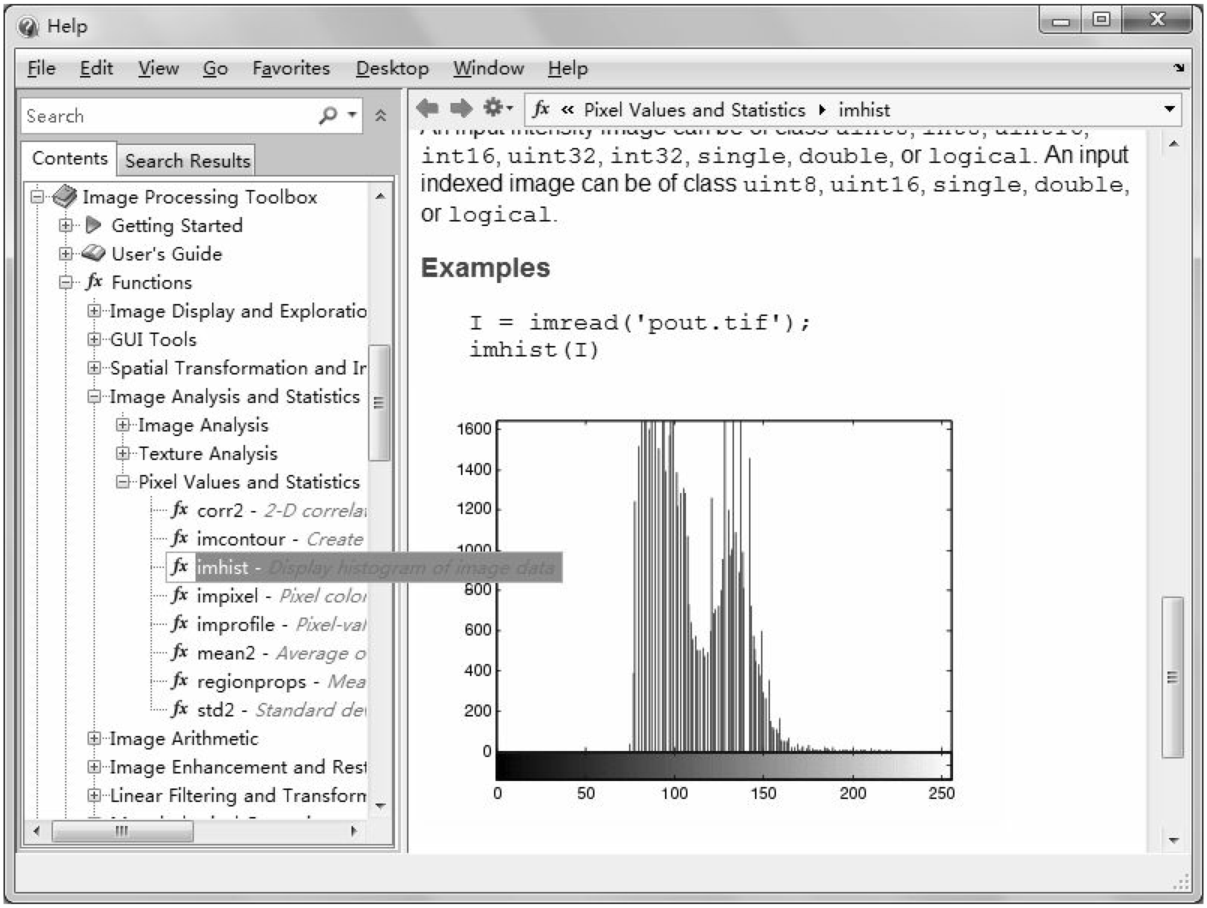
    DOC METHODNAME displays the reference page for the method
    METHODNAME. You may need to run DOC CLASSNAME and use links on the
    CLASSNAME reference page to view the METHODNAME reference page.

    DOC CLASSNAME displays the reference page for the class CLASSNAME.
    You may need to qualify CLASSNAME by including its package: DOC
    PACKAGENAME.CLASSNAME.

    DOC CLASSNAME.METHODNAME displays the reference page for the method
    METHODNAME in the class CLASSNAME. You may need to qualify
    CLASSNAME by including its package: DOC PACKAGENAME.CLASSNAME.

    DOC PRODUCTTOOLBOXNAME displays the documentation roadmap page for
    PRODUCTTOOLBOXNAME in the Help browser. PRODUCTTOOLBOXNAME is the
    folder name for a product in matlabroot/toolbox. To get
    PRODUCTTOOLBOXNAME for a product, run WHICH FUNCTIONNAME, where
    FUNCTIONNAME is the name of a function in that product; MATLAB
    returns the full path to FUNCTIONNAME, and PRODUCTTOOLBOXNAME is
    the folder following matlabroot/toolbox/.
```

▲图2.2 help命令界面



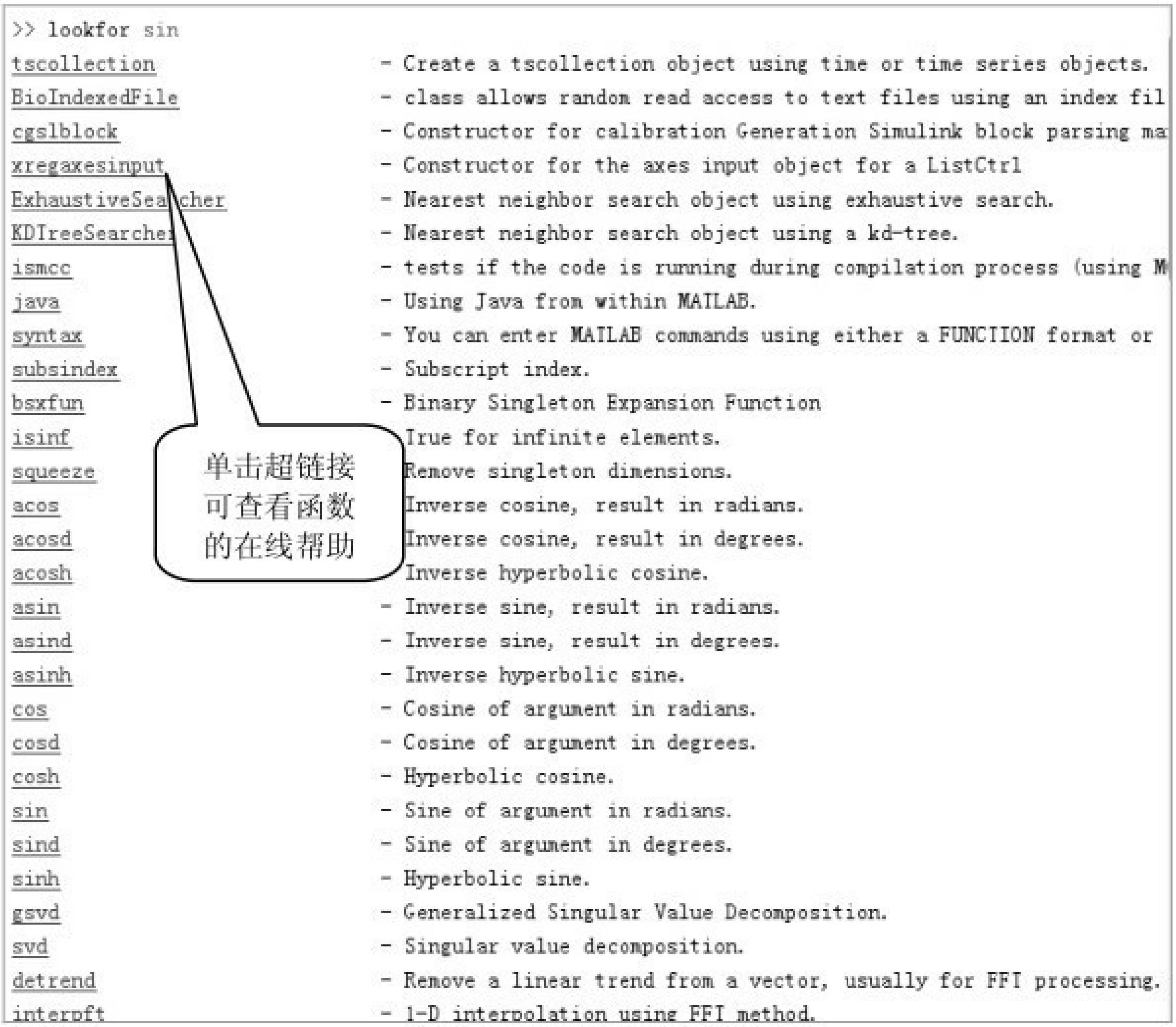
▲图2.3 doc命令结果

3．lookfor命令

当忘记命令或函数的完整拼写时，可以使用lookfor命令查找当前目录和自动搜索列表下所有名字中含有所查内容的函数或命令。其调用格式如下。

lookfor keyword

keyword为指定要查找的关键字。此命令可以给出一个包含指定字符串的函数列表，其中的函数名称为超链接，单击即可查看该函数的在线帮助，如图2.4所示。

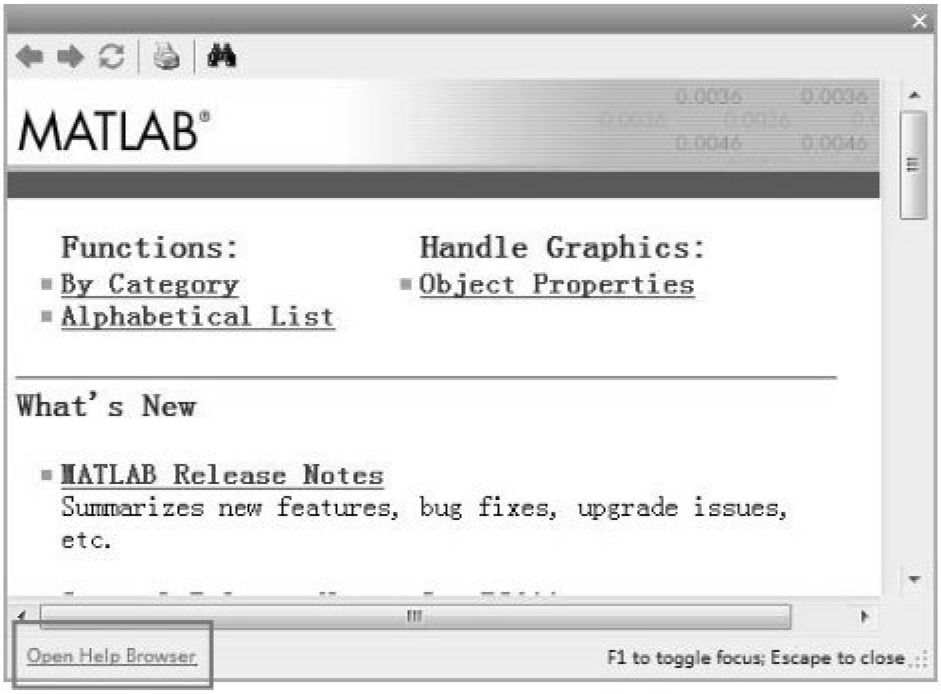


```
>> lookfor sin
tscollection              - Create a tscollection object using time or time series objects.
BioIndexedFile            - class allows random read access to text files using an index fil
cgslblock                 - Constructor for calibration Generation Simulink block parsing ma
xregaxesinput             - Constructor for the axes input object for a ListCtrl
ExhaustiveSearcher        - Nearest neighbor search object using exhaustive search.
KDTreeSearcher            - Nearest neighbor search object using a kd-tree.
ismcc                     - tests if the code is running during compilation process (using M
java                      - Using Java from within MATLAB.
syntax                    - You can enter MATLAB commands using either a FUNCTION format or
subsindex                 - Subscript index.
bsxfun                    - Binary Singleton Expansion Function
isinf                     - True for infinite elements.
squeeze                   - Remove singleton dimensions.
acos                      - Inverse cosine, result in radians.
acosd                     - Inverse cosine, result in degrees.
acosh                     - Inverse hyperbolic cosine.
asin                      - Inverse sine, result in radians.
asind                     - Inverse sine, result in degrees.
asinh                     - Inverse hyperbolic sine.
cos                       - Cosine of argument in radians.
cosd                      - Cosine of argument in degrees.
cosh                      - Hyperbolic cosine.
sin                       - Sine of argument in radians.
sind                      - Sine of argument in degrees.
sinh                      - Hyperbolic sine.
gsvd                      - Generalized Singular Value Decomposition.
svd                       - Singular value decomposition.
detrend                   - Remove a linear trend from a vector, usually for FFT processing.
interpft                  - 1-D interpolation using FFT method.
```


▲图2.4 lookfor命令界面

4．F1命令打开帮助浏览器

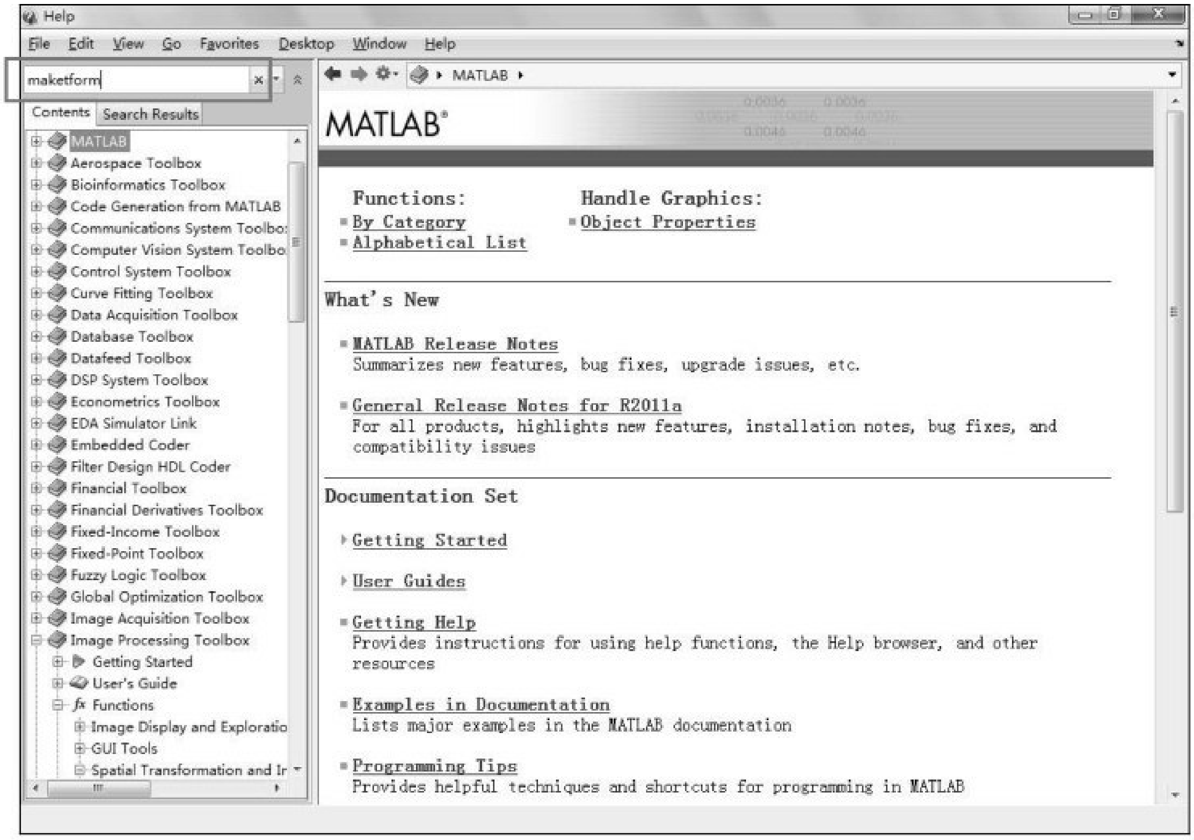
在Matlab R2011a的主界面中按键盘的“F1”键，弹出如图 2.5所示的对话框。



▲图2.5 按 F1键的对话框

单击左下角的“Open Help Browser”链接打开如图2.6所示的帮助浏览器窗口。在左上角的编辑框中输入感兴趣的关键字，单击回车进行查询，右侧会出现相应的帮助信息。

在后面的章节中，如果忘记了曾经提到的命令的含义，建议首先通过在线帮助寻求相关信息，以此增强自学能力。



▲图2.6 Help浏览器窗口

2.1.4 变量的使用

变量可以保存中间结果和输出数值等信息。Matlab中，变量的命名规则和C/C++等常见的编程语言很类似，同时变量也是大小写敏感的。另外，Matlab 中的变量不需要先行定义，但在使用前一定要赋值。

1．变量的赋值

可以通过赋值语句来给变量赋值。赋值操作使用等号“=”，例如a=5是给a（注意不是A ）这个变量赋值5，如果未定义变量a，会自动定义。在Matlab中，变量定义时不需要显式地指明类型， Matlab会根据等号右边的值自动确定变量的类型。默认的对数字的存储类型为double型或double型数组，而字符的存储类型为char型，字符串的存储类型为char

型数组。

对字符串赋值时，需要用半角单引号'括起来（注意不是双引号，也不是任何的全角字符），例如msg='Hello world'。

2．内部变量

Matlab有某些内部变量名和保留字，如表2.2所示。变量命名时不要与它们重名。

表2.2 Matlab内部变量列表

特殊变量	说明
ans	默认的结果输出变量
pi	圆周率
Inf 或 inf	无穷大值，如 1/0
i 和 j	单位虚数值
eps	浮点运算的相对精度

续表

特殊变量	说明
realmax	最大的正浮点数
realmin	最小的正浮点数
NaN 或 nan	不定量，如 0/0
nargin	函数输入参数个数
nargout	函数输出参数个数
lasterr	最近的错误信息
lastwarning	最近的警告信息
computer	计算机类型
version	Matlab 版本

3．查看工作区中的变量

使用who和whos命令可以查看所有当前工作区中变量的情况。使用clear或clear all命令可以清除工作区中所有的变量定义，也可以在clear后面加上变量名清除特定的变量定义。另外，clc命令可以用来清屏，所以

这两个命令常常用在M文件的开头用来构造一个干净的工作区。

```
>> a = 1; %定义一个数值型变量a
>> str = 'hello'; %定义一个字符串变量（字符数组）
>> v = [3 2 1] %定义一个数值型向量
v =
      3    2    1
>> whos
Name    Size          Bytes Class    Attributes
a          1×1            8 double
str       1×5            10 char
v          1×3            24 double
>> clear all
>> whos
>>
```

4．数据类型及其转换

Matlab中的数据类型列表如表2.3所示。

表2.3 Matlab数据类型

数据类型	说　　明
double	Matlab 中最常见也是默认的数据类型，双精度方式存储的浮点数。有效范围是-10^{308}～10^{308}。这同时也是 Matlab 所能直接给出的最大数值范围。此种类型占用的内存空间为 8 字节
uint8	8 位无符号整数，范围是 0～255。此种类型占用的内存空间为 1 字节
uint16	16 位无符号整数，范围是 0～655 35。此种类型占用的内存空间为 2 字节
uint32	32 位无符号整数，范围是 0～4 294 967 295。此种类型占用的内存空间为 4 字节
Uint64	64 位无符号整数，范围是 0～18 446 744 073 709 551 615。此种类型占用的内存空间为 8 字节
int8	8 位有符号整数，范围是-128～127。此种类型占用的内存空间为 1 字节
int16	16 位有符号整数，范围是-32 768～32 767。此种类型占用的内存空间为 2 字节

续表

数据类型	说　明
int32	32 位有符号整数，范围是-2 147 483 648～2 147 483 647。此种类型占用的内存空间为 4 字节
int64	64 位有符号整数，范围是-9 223 372 036 854 775 808～9 223 372 036 854 775 807。此种类型占用的内存空间为 8 字节
single	单精度浮点数，范围是-10^{38}～10^{38}。此种类型占用的内存空间为 4 字节
char	字符型变量，占用空间为 2 字节
logical	布尔型变量，占用空间为 1 字节。此种类型的转换函数也可以使用 boolean 与 logical 等效

默认情况下，Matlab将变量存储为双精度浮点数（double），而Matlab中的很多函数也只接受这种类型的数据。然而，图像处理操作中经常使用到uint8等类型的数据，这就需要执行数据类型的强制转换操作。这种操作很简单，调用格式统一如下。

Destination_Var = type_name(Source_Var)

其中，type_name即数据的存储类型，Destination_Var和Source_Var分别为目标变量和原始变量。例如下面的命令将double原始变量a转换为uint8变量b。

```
>> a = 1;
>> b = uint8(a);
```

5．读取与保存工作区中的变量

save命令可以将当前工作区的变量以二进制的方式保存到扩展名为MAT的文件中；load命令可以读出这样的文件。它们的调用格式如下。

save filename arg1 arg2 arg3, …

load filename arg1 arg2 arg3, …

● filename参数指定保存或读取变量所使用的文件名。如果不指定文件名，默认使用的文件是Matlab.MAT。

● arg1、arg2、arg3等参数是需要从文件中存储或读出的变量名。这两个命令分别可以存储或读取一个或一组变量。

下面的命令将price、age和number三个变量保存到文件MyData.mat中。

```
>> save('MyData.mat', 'price', 'age', 'number')
```

提示

也可以不指定变量名，从而将当前工作区中所有的变量一起储存到mat 文件或将文件中保存的所有变量一起读入工作区，这个批量保存和读取功能在运行非常耗时的程序时显得十分有用——由于Matlab执行效率并不高（和Visual C++相比），所以对于一个计算量很大的程序而言，运行几个小时并不稀奇。这时，可以根据需要在希望中断程序时保存程序的所有上下文变量，以备之后随时从中断点开始执行。

2.1.5 矩阵的使用

1．矩阵的定义

在 Matlab 中定义矩阵很简单。可以使用半角分号分隔行与行，使用半角逗号（或者空格）分隔列与列来直接定义矩阵，例如下面的命令就定义了一个3行3列的二维矩阵A。

```
A=[1, 2, 3; 4, 5, 6; 7, 8, 9]
A =
    1    2    3
    4    5    6
    7    8    9
```

还有另一种方式可以生成行向量，[begin:inc:end]会生成从begin开始到end结束，增量为incre的一系列数字组成的向量。例如v=[2:1:10]表示生成从2到10的间隔为1的向量（一维矩阵），即

```
v= [2:1:10]
v=
    2    3    4    5    6    7    8    9    10
```

如果间隔为1，也可以忽略中间的参数，直接输入I=[2:10]即可。

2．生成特殊矩阵

除直接定义外，还可以通过函数生成特定的矩阵，例如eye(n)生成N 阶单位阵，zeros(n)生成N阶每个元素均为0的方阵，magic(n)生成N阶幻方阵等等。常见的用于生成矩阵的函数列表如表2.4所示。

表2.4 生成矩阵的函数

函数名称	用　　途
eye	产生单位矩阵
zeros	产生全部元素为 0 的矩阵
ones	产生全部元素为 1 的矩阵
true	产生全部元素为真的逻辑矩阵
false	产生全部元素为假的逻辑矩阵
rand	产生均匀分布随机矩阵
randn	产生正态分布随机矩阵
randperm	产生随机排列
Linspace	产生线性等分的矩阵
Logspace	产生对数等分向量
Company	产生伴随矩阵
Hadamarb	产生 Hadamarb 矩阵
Magic	产生幻方矩阵
Hilb	产生 Hilbert 矩阵
Invhilb	产生逆 Hilbert 矩阵

3．获得矩阵大小和维度

size函数可以获得指定数组某一维的大小，可以用来查看图像的高度和宽度以及动态图像的帧数等。其调用方法如下。

size(A,dim)

● A为需要查看大小的数组。

● dim为指定的要查看的维数，这是一个可选参数，若不指定此参数，返回值为一个包含数组从第一维到最后一维大小的数组。

例如，对于一个3行5列的矩阵B，有 size(B, 1)=3，size(B, 2) = 5，

size(B) = [3 5]。

函数ndims可以查看数组的维数。调用方式如下。

ndims(A)

其中A为需要查看维数的数组。

4．访问矩阵元素

访问矩阵的一个元素的方式是在矩阵名字的后面注明行列序号，例如访问 A 的第三行第二列元素就是 A(3,2)。提取矩阵的一整行元素，如要提出 A 的第二行使用 A(2,:)，如果是第二列则是A(:,2)；而A(:)表示将矩阵按列存储得到一个长列向量。示例如下。

```
>> A=[1, 2, 3; 4, 5, 6; 7, 8, 9] ; %定义矩阵A
>> A(1, :) %提取第1行
ans =
1    2    3
>> A(:, 3) %提取第3列
ans =
     3
     6
     9
>> A(:)'
ans =
     1    4    7    2    5    8    3    6    9
```

注意

Matlab中的矩阵下标是从1开始的。对图像矩阵也是一样，所以一个m×n的矩阵实际的下标范围为[1:m]和[1:n]。

对于矩阵A，提取矩阵元素或子块的方法如表2.5所示。

表2.5 提取矩阵元素或子块的方法

命令片断	用途
A(m,n)	提取 *m* 行 *n* 列位置的一个元素
A(:,n)	提出第 *n* 列
A(m,:)	提出第 *m* 行
A(m1:m2, n1:n2)	提出 *m*1 到 *m*2 行，*n*1 到 *n*2 列的一个子块
A(m:end, n)	提出 *m* 行到最后一行，第 *n* 列的一个子块
A(:)	将矩阵按列存储得到一个长列向量

5．进行矩阵运算

可以像对数字操作一样对矩阵进行操作，常见算术运算符的使用方法如表2.6所示。

表2.6 常见的算术运算符

运算	符号	对应函数	说明
加	+	plus(A,B)	
减	-	minus(A,B)	
乘	*	mtimes(A,B)	即通常意义上的矩阵乘法

续表

运算	符号	对应函数	说明
点乘	.*	times(A,B)	矩阵的对应元素相乘。参与运算的两个矩阵必须拥有同样的大小
乘方	.^	mpower(A,B)	对矩阵的每一个元素进行指定幂次的乘方
矩阵乘方	^	power(A,B)	
矩阵左除	\	mldevide(A,B)	左除 A\B 相当于 inv(A) * B
矩阵右除	/	mrdevide(A,B)	右除 A/B 相当于 B * inv(A)
左除	.\	ldevide(A,B)	矩阵中对应位置的元素的左除
右除	./	rdevide(A,B)	矩阵中对应位置的元素的右除
矩阵与向量转置	.'	transpose(A,B)	这里的转置不对复数进行共轭操作
复数矩阵转置（共轭）	'	ctranspose(A,B)	应用于复数数值时的含义是取共轭，应用于实数矩阵时的含义与普通转置相同，应用于复数矩阵时首先对所有元素取共轭再求矩阵转置

矩阵运算的求值顺序和一般的数学求值顺序相同：表达式是从左向

右执行的，幂运算的优先级最高，乘除次之，最后是加减。如果有括号，那么括号的优先级最高。

对于图像矩阵，还有一系列Matlab函数可以进行专门针对图像的像素级操作。如图像叠加imadd，图像相减imsubtract等。

2.1.6 细胞数组和结构体

1．细胞数组

在处理函数返回值和示波器部件输出时，常常会遇到不同维度的返回值同时被一个函数返回的情况。同时，通常也希望能使函数的输入参数尽可能少。Matlab提供了允许这样做的方式。

细胞数组是Matlab特有的一种数据结构，它的各个元素可以是不同的数据类型。细胞数组可采用下标访问。

例如，一个细胞数组可以采用如下方式定义。

```
Cell = {'Harry', 15, [1 0; 15 2]};
```

也可以通过{}加上索引来直接定义细胞数组的某个元素，如下所示。

```
%定义细胞数组的另一种方式
>>Cell {1}= 'Harry';
>>Cell{2}= 15;
>>Cell{3}= [1 0; 15 2];
```

注意使用花括号{}而不是方括号[]来定义细胞数组。对细胞数组的访问方式也很简单，同样使用花括号{}来给定索引值。

```
%访问细胞数组
>> Cell{1}
ans =
Harry
```

```
>> Cell{2}
ans =
    15
>> Cell{3}
ans =
    1 0
    15 2
```

而使用圆括号形式的索引可以得到变量的描述，如下所示。

```
>> Cell(3)
ans =
    [2x2 double]
```

注意

细胞数组中存储的是建立该对象时所使用的其他对象（矩阵或字符串、数字等）的拷贝，而不是引用或指针，即使其他对象的值被改变，细胞数组中的值也不变。

2．结构体

结构体是另一种形式的聚合类型，它与C/C++中的结构体或类很相似，拥有多个不同类型的字段，通过圆点运算符“.”引用内部字段，字段必须具有独特的名字以便区分。访问结构体的方式与定义的方式相同。上面的例子如果用结构体表示，则为如下所示。

```
% 定义结构体
Struct.Name = 'Harry';
Struct.Age = 15;
Struct.SalaryMatrix = [1 0; 15 2];
>>
>> Struct %显示结构体的内容
Struct =
```

```
Name: 'Harry'
Age: 15
SalaryMatrix: [2×2 double]
>>
% 访问结构体的内部字段
>>name = Struct.Name;
```

而访问结构体内容时，使用相同的语法即可，例如Struct.Name的值仍然是Harry。

这两种复合类型在保存用户输入和使用Simulink仿真输出时尤为常用。

2.1.7 关系运算与逻辑运算

关系运算符的运算结果是布尔量（0或1），具体说明如表2.7所示。

表2.7 关系运算符的使用

运　算	符　号	运　算	符　号
大于	>	小于	<
大于等于	>=	小于等于	<=
等于	==	不等于	~=

Matlab同样支持逻辑运算，常见的逻辑运算符如表2.8所示。

表2.8 逻辑运算符的使用

运　算	符　号	运　算	符　号
与	&	或	|
非	~	异或	Xor

2.1.8 常用图像处理数学函数

Matlab最为强大的功能是依靠函数实现的，这些函数可能是Matlab内置的，也可能是由M文件提供的。常见的有sin、cos、tan、log、log2这样的数值函数和trace这样的矩阵函数，还有逻辑函数等。逻辑函数和矩阵函数在图像处理中应用较多，表2.9所列为其中较常用的一部分函数。关于这些函数更详细的用法描述，可以通过help或open命令获得。

表2.9 常用的逻辑函数和矩阵函数

函　数	用　途
all	是否所有元素非零
any	是否至少有一元素非零
isempty	是否空矩阵
isequal	是否两矩阵相同
isinf	判断有无 inf 元素
isnan	判断有无 nan 元素
isreal	判断是否实矩阵
find	返回一个由非零元素下标组成的矩阵
det	计算方阵对应的行列式值
diag	抽取对角线元素
eig	求特征值和特征向量
fliplr	左右翻转
flipud	上下翻转
inv	求逆矩阵
lu	三角分解
norm	求范数
orth	正交化
poly	求特征多项式
qr	正交三角分解
rank	求矩阵的秩
svd	奇异值分解
trace	求矩阵的迹
Tril	抽取上三角阵
Tnu	抽取下三角阵

调用Matlab函数的方法如下。函数名后使用一对圆括号括住提供给函数的参数，如sin(t)。如果函数有返回值，但调用者没有指定接收返回值的变量，系统会使用默认的ans变量存储返回值。如果函数返回多个值，则ans中只保留第一个返回值，因此对于这种情况应显式地使用向量来接收返回值，如下所示。

[V D] = eigs(A) ; %计算矩阵A的特征值和特征向量，返回值中V为特征向量，D为特征值

使用函数还应当注意几点。

（1）函数只能出现在等式的右边。

（2）每个函数依原型不同，对自变量的个数和类型有一定的要求，如sin和sind函数。

（3）函数允许按照规则嵌套，比如sin(acos(0.5))。

2.1.9 Matlab程序流程控制

Matlab 提供了程序流程分支控制的语句，它们的用法和 C/C++中几乎完全一致，如表 2.10所示。

表2.10 程序流程控制方法

语　　句	规 范 写 法	备　　注
if…elseif…else	if expression1 　statements1 elseif expression2 　statements2 else expression3 　statements3 end	如果 elseif 使用的层次较多，可以考虑转而使用 switch
for	for index=start:increment:end 　statements end	increment 指定步进值，省略则默认为 1。可以嵌套使用
while	while expression 　statements end	同样可以嵌套使用
break	-	终止 while 或 for 循环的执行
continue	-	直接跳到下一个循环
switch	switch expression 　case expression1 statements1 　case expression2 　　statements2 　otherwise statements_other end	没有默认的 fall-through，因而不需要使用配套的 break 语句
return	-	返回调用函数

1．简要示例

在后面将多次使用这些语句，因而，在此给出几个简单的例子。读者可以在金羽图书论坛（http://bbs.book95.com）的“金羽图书与答疑”板块与本书同名的主题帖子附件中的 chapter2/code目录下找到示例2.1～2.3所对应的m文件，也可以通过菜单**“File”**→**“New”**→**“M-Files”**新建一个M文件，在出现的窗口中粘贴这些代码，然后运行。在非注释行处按下F12可以设置断点，按下F5键可以运行程序。

[例2.1]if语句和for循环及其嵌套。

```
%ex2_1.m
arg=input('Input argument:');    % 提示输入arg变量
total = 0; detail = 0;
% if语句开始
```

```
if(arg==1)
    % 外层for语句开始
    for i=1:1:5
      total = total + 1;
      % 内层for语句开始
      for j=1:0.1:2
        detail = detail + total;
      % 内层for语句结束
      end
    % 外层for语句结束
  end
% if语句的另一分支
elseif (arg=2)
      total = 0;
      detail = total;
% if语句的其他所有分支
else
        error('Invalid arguments!');
% if语句结束
end
detail    % 显示detail变量
```

（请注意本例中分号的使用）

[例2.2]与例2.1类似的功能，使用switch分支和while循环。

```
%ex2_2.m
arg=input('Input argument:');
total = 0; detail = 0;
% switch语句开始
```

```
switch arg
      % 分支1
      case 1
        i=1;
        % 外层while语句开始
        while (i<=5)
          total = total + 1;
          i = i + 1;
          j = 1;
          % 内层while语句开始
          while (j<=2);
            detail = detail + total;
            j = j + 0.1;
          % 内层while语句结束
          end
        % 外层while语句结束
        end
      % 分支2
      case 2
        total = 0;
        detail = total;
      % 分支其他
      case others
        error('Invalid arguments');
% switch语句结束
end
detail
```

总结这两个例子，可以发现，在分支较多时使用switch是合算的，而for和while用于循环控制，这一点与C/C++是完全相同的。但是，相对于C/C++，Matlab有一个突出的优点，就是可以自动生成元素之间具有特定间隔的矩阵，从而避免使用某些循环，这里仅仅使用二维的情况举例。

[例 2.3]产生一幅亮度按对角线方向的余弦规律变化的灰度图，比较一维方法和二维方法所需的时间。

```
A = rand(3000, 3000);
f = zeros(3000, 3000);
u0 = 100; v0 = 100;
tic; % 开始计时
% 一维方法
% 外层for循环开始
for r=1:3000
        u0x=u0*(r-1);
        % 内层for循环开始
        for c=1:3000
          v0y=v0*(c-1);
          f(r,c) = A(r,c) * cos(u0x+v0y);
        % 内层for循环结束
        end
% 外层for循环结束
end
t1=toc    % 停止计时并记录时间到t1
tic;    % 重新开始计时
% 二维方法
r = 0:3000-1;
```

c = 0:3000-1;

[C, R] = meshgrid(c, r);

% meshgrid是生成网格坐标的函数，实际就是生成需要的二维像素点的坐标拟合表示。

% 建议读者在这里中断，观察一下C和R矩阵的内容。

g = A .* cos(u0 .* R + v0 .* C);

%系统将自动执行“循环”操作，实质是对R和C中每个数据按照指定公式操作

t2 = toc % 停止计时并将计时值保存到t2

在运行之后，我们发现，t2远小于t1。因此，在用Matlab对数字图像按像素进行操作时，需要尽可能地避免使用笨拙的多层嵌套循环。

2．meshgrid函数

例2.3中用到的meshgrid函数用于根据给定的横纵坐标点生成坐标网格，以便计算二元函数的取值，在绘制三维曲面时常常会用到它。其调用方式如下。

[X,Y] = meshgrid(x, y)

参数说明

● x为输入的横坐标。

● y为输入的纵坐标。

返回值

● X和Y为输出采样点的横坐标矩阵和纵坐标矩阵，X阵和Y阵的元素分别为对应位置点的横坐标和纵坐标。

下面以绘制二维高斯函数曲面为例说明meshgrid的用法。

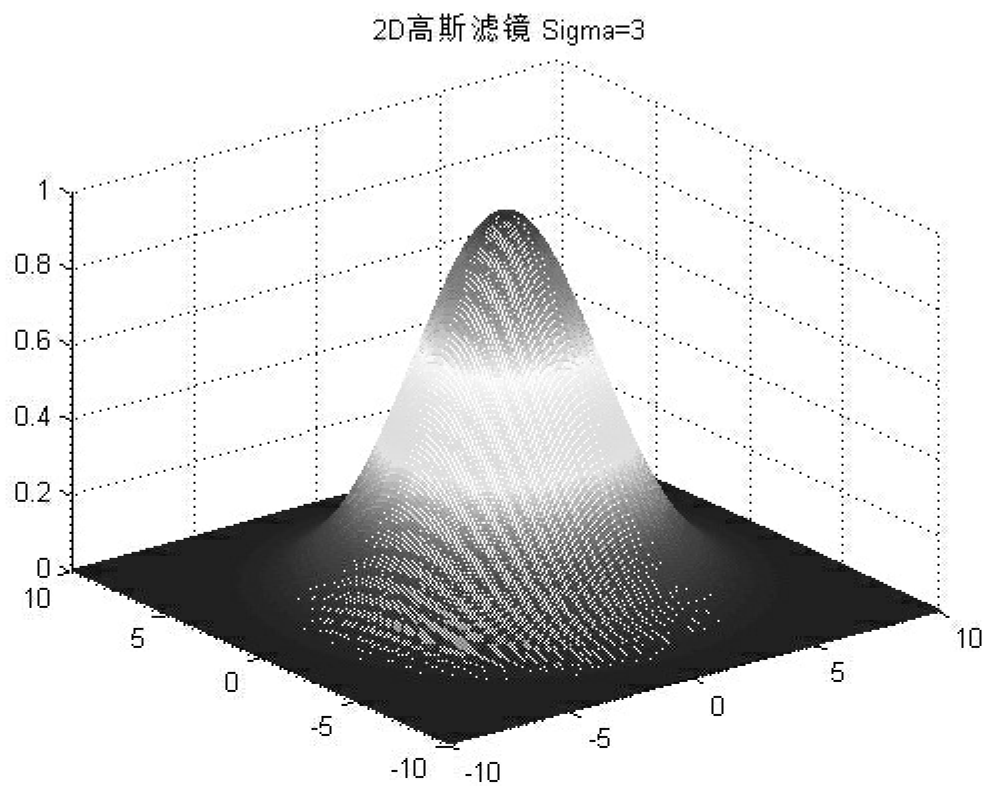
中心在原点的二维高斯函数表达式如下。

$$H(u,v) = \mathrm{e}^{-[u^2+v^2]/2\sigma^2}$$

下面的程序分别为u和v赋值[-10:0.1:10]，令σ=3，使用meshgrid函数生成网格，并计算函数值，（注意这里使用的是.^和./，而不是^和/，因为计算的对象是矩阵中的元素），然后再使用 mesh函数将其显示到绘图窗口中。

```
u = [-10:0.1:10];
v = [-10:0.1:10];
[U,V] = meshgrid(u,v);
H = exp(-(U.^2 + V.^2)./2/3^2);
mesh(u, v, H);
% mesh函数是绘制三维曲面的函数，第一个和第二个参数分别为 x轴和 y轴的坐标点序列，第三个参数为在由坐标
点序列确定的每一个方格点上的函数值
```

生成的图像如图2.7所示。



▲图2.7 高斯函数曲面

优化小技巧

提前分配矩阵内存

这个技巧与动态内存的使用有关。在 C/C++里，使用大块动态内存往往意味着堆操作，而当分配的动态内存零散无序时，会产生大量内存碎片，进而导致内存分配和回收效率降低。所以，可以事先分配一块足够大的空间（当然，不是过大）以尽量减少内存碎片的产生。事实上，Matlab中分配动态内存远没有C/C++那样麻烦，只需要类似如下一条语句即可。

```
memo = zeros(1024, 128);
```

这条语句本来是用于构造一个元素全部为零的矩阵，但同时很自然地也就分配了一块足够大的空间。

2.1.10 M文件编写

M文件和C/C++中c/cpp文件类似，就是存储Matlab代码并可以执行的文件。Matlab的源代码文件可以直接执行而不需编译（也可以通过编译来使代码运行得更快）。很多情况下，M文件用于封装一个功能函数从而提供某些特定功能。一般来说，M文件以文本格式存储，执行顺序从第1行开始向下，遇到终止语句结束，用户可以在M文件中定义函数和过程。

M文件可以使用任何文本编辑器编写，但最常使用的还是Matlab自带的M文件编辑器。如果是编写函数，最好将它放在Matlab的搜索路径列表中的某个目录下，并与系统自带的M文件分开，以便管理。

对于位于当前工作目录中的M文件，可以直接在命令行输入其文件名来运行它。作为函数时，M文件也可以接受参数。

稍后将提供一个Matlab自带M文件的例子，并给予简单分析。

2.1.11 Matlab函数编写

1. 函数语法

Matlab函数通常在M文件中定义，一个文件可以定义多个函数。一个Matlab函数通常包含以下组成部分。

● 函数定义行

function [outputs] = name(inputs)

Matlab允许返回多个参数（outputs），如果只返回一个参数，可以省略方括号。需要注意的是，输入参数是使用圆括号括起来的。例如，如果要定义一个用于平滑图像的函数imsmooth，可以将定义行书写如下。

function [imgOut, retCode] = imsmooth(imgIn, args)

某些函数可能没有输出参数，那么就需要在省略方括号及其中内容的同时，省略等号。于是无返回值的函数就需要定义为如下所示。

function imsmooth(imgIn, args)

Matlab允许区分函数名的前63个字母，多出的字母将被忽略。函数的命名规则与C/C++类似，必须以字母开头，可以包含字母、数字和下划线，但不能包含空格。

函数可以在其他的M函数中被调用，也可以在命令行直接调用。调用函数的方法很简单，只需要写出函数定义中除了function之外的部分即可。示例如下。

[a,b] = imsmooth(I,arg);

● “H1”行

“H1”行是M文件中的第一个注释行（即以百分号开始的行），它必须紧跟着函数定义行，中间不能有空行，这一行的百分号前也不能有空白字符或缩进。这一行的内容将在使用help命令时显示在第一行，而lookfor命令查找H1行中的指定关键词，并在结果的右侧列显示H1行。

一个典型的H1行的例子如下。

% IMSMOOTH Perform smooth operation on specified image with certain arguments.

这样，smooth函数在lookfor命令查找的时候就会显示如下。

IMSMOOTH　SMOOTH Perform smooth operation on specified image with certain arguments.

● 帮助文本

帮助文本的位置和约定与H1行类似，只能紧跟H1行，中间不能有任何空行或者缩进。同样，帮助文本的本质就是注释行，因而需要以%开头。

Matlab通过判断是否紧跟函数定义来判断一个注释行究竟是H1行、帮助文本还是普通注释。因此可以在加入一个或多个空行后加入普通注释，使用HELP命令将不会显示普通注释的内容。

● 函数体和备注

这些部分的编写方式和普通的 Matlab 程序类似，如果有返回值，应在函数体中为输出变量赋值。

2．一个**M**文件的例子片断 **– imfinfo.m**

（可以在命令行输入 edit imfinfo查看完整源文件）

function info = imfinfo(filename, format)　**%**函数定义

%IMFINFO Information about graphics file.　**%H1**行

%　INFO = IMFINFO(FILENAME,FMT) returns a structure whose **%**帮助文本

%　fields contain information about an image in a graphics

%　file. FILENAME is a string that specifies the name of the

%　graphics file, and FMT is a string that specifies the format

%　of the file. The file must be in the current directory or in

%　a directory on the Matlab path. If IMFINFO cannot find a

%   file named FILENAME, it looks for a file named
FILENAME.FMT.

%

%   The possible values for FMT are contained in the file format

%   registry, which is accessed via the IMFORMATS command.

%

%   If FILENAME is a TIFF, HDF, ICO, GIF, or CUR file containing
more

%   than one image, INFO is a structure array with one element for

%   each image in the file. For example, INFO(3) would contain

%   information about the third image in the file.

%

%   INFO = IMFINFO(FILENAME) attempts to infer the format of the

%   file from its content.

%

%   INFO = IMFINFO(URL,...) reads the image from an Internet URL.

%   The URL must include the protocol type (e.g., "http://").

%

%   The set of fields in INFO depends on the individual file and

%   its format. However, the first nine fields are always the

%   same. These common fields are:

%

%   Filename   A string containing the name of the file

%

%   FileModDate   A string containing the modification date of

%   the file

%

```
%   FileSize    An integer indicating the size of the file in
%                    bytes
%
%   Format    A string containing the file format, as
%                 specified by FMT; for formats with more than one
%                 possible extension (e.g., JPEG and TIFF files),
%                 the first variant in the registry is returned
%
%   FormatVersion    A string or number specifying the file format
%                          version
%
%   Width    An integer indicating the width of the image
%              in pixels
%
%   Height    An integer indicating the height of the image
%                 in pixels
%
%   BitDepth    An integer indicating the number of bits per
%                    pixel
%
%   ColorType    A string indicating the type of image; this could
%                    include, but is not limited to, 'truecolor' for a
%                    truecolor (RGB) image, 'grayscale', for a grayscale
%                    intensity image, or 'indexed' for an indexed image.
%
%   If FILENAME contains Exif tags (JPEG and TIFF only), then the
INFO
```

```
%    struct may also contain 'DigitalCamera' or 'GPSInfo' (global
%    positioning system information) fields.
%
%    The value of the GIF format's 'DelayTime' field is given in
hundredths
%    of seconds.
%
%    Example:
%
%       info = imfinfo('ngc6543a.jpg');
%
%    See also IMREAD, IMWRITE, IMFORMATS.
%    Copyright 1984-2008 The MathWorks, Inc.
%    $Revision: 1.1.6.14 $ $Date: 2009/11/09 16:27:13 $
error(nargchk(1, 2, nargin, 'struct'));    %函数体和注释
…（函数体）
% Delete temporary file from Internet download.
if (isUrl)
    deleteDownload(filename);
end
```

2.2 Matlab图像类型及其存储方式

在1.1.3小节介绍数字图像的分类时，曾接触到一些主要的图像类型。本节我们就来看一看这些主要的图像类型在Matlab中是如何存储和表示的，主要包括亮度图像、RGB图像、索引图像、二值图像和多帧图像。

1．亮度图像（**Intensity Image**）

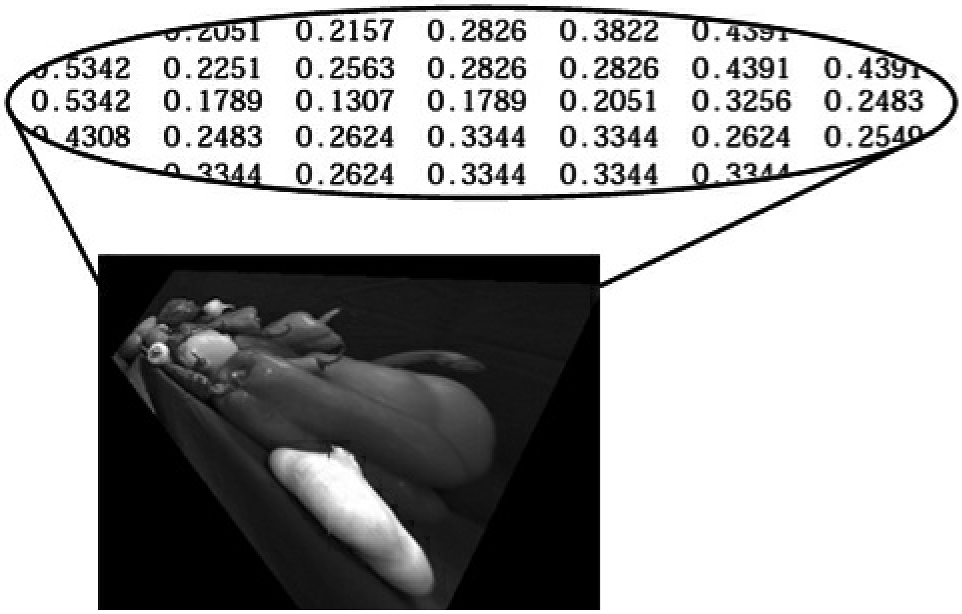
亮度图像即灰度图像。Matlab 使用二维矩阵存储亮度图像，矩阵中的每个元素直接表示一个像素的亮度（灰度）信息。例如，一个200像素×300像素的图像被存储为一个200行300列的矩阵，可以使用 2.1.5 小节介绍的选取矩阵元素（或子块）的方式来选择图像中的一个像素或一个区域。

如果矩阵元素的类型是双精度的，则元素的取值范围是 0～1；如果是 8 位无符号整数，则取值范围0～255。数据0表示黑色，而1（或255）表示最大亮度（通常为白色）。

图2.8所示是一个使用双精度矩阵存储亮度图像的例子。

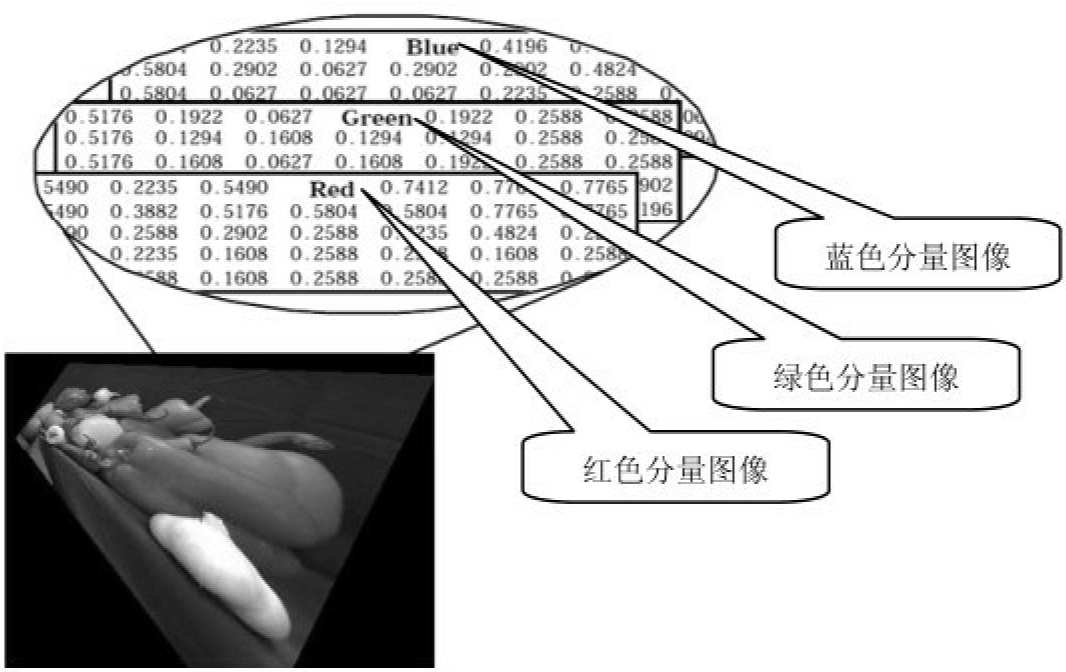
2．**RGB**图像（**RGB Image**）

RGB图像使用3个一组的数据表达每个像素的颜色，即其中的红色、绿色和蓝色分量。在Matlab中，RGB图像被存储在一个m×n×3的三维数组中。对于图像中的每个像素，存储的三个颜色分量合成像素的最终颜色。例如，RGB图像I中位置在11行40列的像素的RGB值为I(11,40,1:3)或I(11,40,:)，该像素的红色分量为I(11,40,1)，蓝色分量为I(11,40,3)。而I(:,:,1)则表示整个的红色分量图像。



▲图2.8 Matlab中亮度图像的表示方法

RGB图像同样可以由双精度数组或8位无符号整数数组存储。图2.9所示是一个使用双精度数组存储RGB图像的例子。



▲图2.9 Matlab中 RGB图像的表示方式

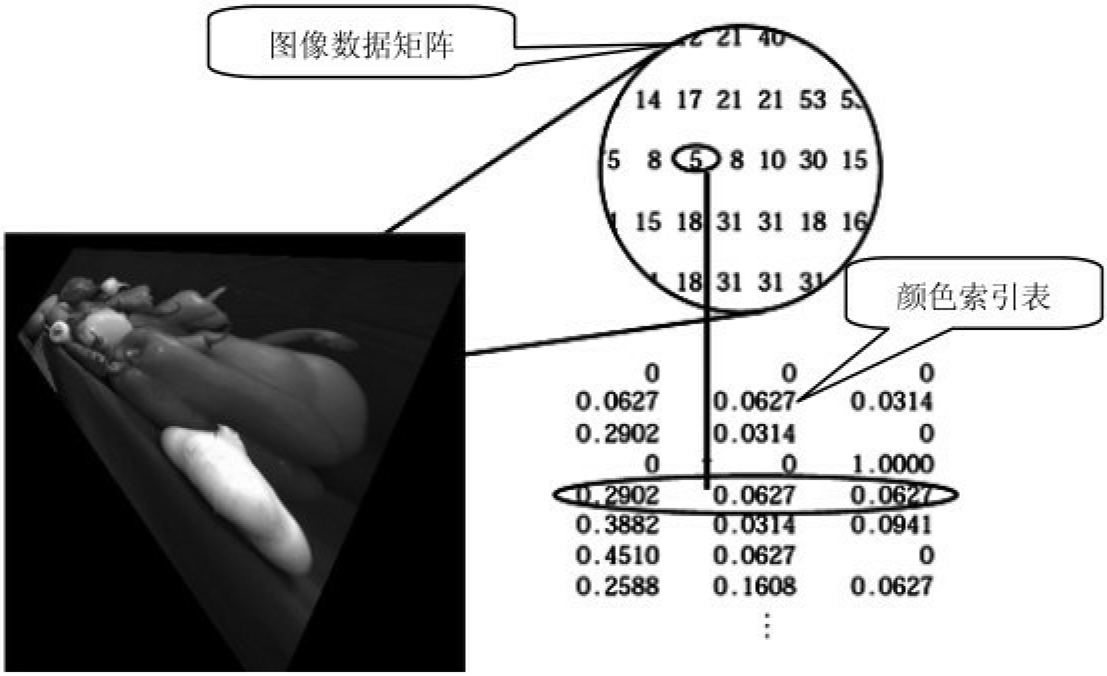
3．索引图像（**Indexed Image**）

索引图像往往包含两个数组：一个图像数据矩阵（Image Matrix）和一个颜色索引表（Colormap）。对应于图像中的每一个像素，图像数据数组都包含一个指向颜色索引表的索引值。

颜色索引表是一个m×3的双精度型矩阵，每一行指定一种颜色的三个RGB分量，即 color = [R G B]。其中R、G、B是实数类型的双精度数，取值0～1。0表示全黑，1表示最大亮度。图 2.10给出了一个索引图像的实例，注意图像中的每个像素都用整数表示，其含义为颜色索引表中对应颜色的索引。

图像数据矩阵和颜色索引表的关系取决于图像数据矩阵中存储的数据类型是双精度类型还是8位无符号整数。

如果图像数据使用双精度类型存储，像素数据1表示颜色索引表中的第一行，像素数据2表示颜色索引表中的第二行，以此类推。而如果图像数据使用8位无符号整数存储，则存在一个额外的偏移量-1，像素数据0表示颜色索引表中的第一行，而1表示索引表中的第二行，以此类推。



▲图2.10 Matlab中索引图像的表示方法

8位方式存储的图像可以支持256种颜色（或256级灰度）。图2.10中，数据矩阵使用的是双精度类型，所以没有偏移量，数据5表示颜色表中的第5种颜色。

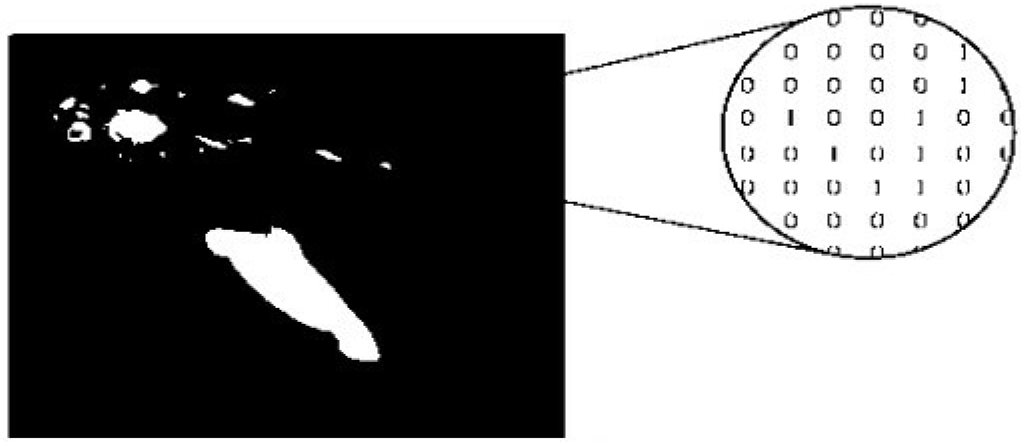
4．二值图像（**Binary Image**）

在二值图像中，像素的颜色只有两种可能取值：黑或白。Matlab将二值图像存储为一个二维矩阵，每个元素的取值只有0和1两种情况，0表示黑色，而1表示白色。

二值图像可以被看做是一种特殊的只存在黑和白两种颜色的亮度图像，当然，也可以将二值图像看做是颜色索引表中只存在两种颜色（黑和白）的索引图像。

Matlab中使用uint8型的逻辑数组存储二值图像，通过一个逻辑标志表示数据有效范围是0～1，而如果逻辑标志未被置位，则有效范围为0～255。

二值图像的表示方法如图2.11所示。



▲图2.11 Matlab中二值图像的表示方法

5．多帧图像（**Multiframe Image Array**）

对于某些应用，可能要处理多幅按时间或视角方式连续排列的图像，称之为多帧图像（所谓“帧”就是影像动画中最小单位的单幅影像画面），如核磁共振成像数据或视频片断。Matlab提供了在同一个矩阵中存储多帧图像的方法，实际上就是在图像矩阵中增加一个维度来代表时间或视角信息。例如，一个拥有5张连续的400像素×300像素的RGB图像的多帧连续片断的存储方式是一个400×300×3×5的矩阵，一组同样大小的灰度图像则可以使用一个400×300×1×5的矩阵来存储。

如果多帧图像使用索引图像的方式存储，只有图像数据矩阵被按多帧形式存储，而颜色索引表只能公用。因此，在多帧索引图像中，所有的索引图像公用一个颜色索引表，进而只能使用相同的颜色组合。

● cat函数

cat函数可以在指定维度上连接数组，其调用方式如下。

CAT(DIM, A, B);

或

CAT(DIM, A1, A2, …);

此函数在第DIM维度将第2至第n个参数提供的数组连接起来。于是，若要构造一个由5幅RGB图像构成的多帧图像组，使用的命令如下。

ANIM=CAT(4, A1, A2, A3, A4, A5);

● 选择存储方式时的限制

图像处理工具箱中的某些函数只能处理图像矩阵中的前2维或前3维信息。当然，也可以使用它们处理拥有4个维度或5个维度的RGB图像或者连续图像序列，但这需要单独处理每帧符合要求的亮度/二值/索引/RGB图像。例如，显示ANIM中的第3帧图像需要使用如下方式。

imshow(ANIM(:,:,:,3));

函数imshow作用是显示一帧图像，详见2.5节。

注意

如果向一个函数传递了超过其所能够处理的维度的图像矩阵，那么结果可能是不确定的。某些函数的行为可能是处理图像的第一帧或第一个颜色维度，但某些函数可能带来不确定的行为和处理结果。

默认情况下，Matlab将绝大多数数据存储为双精度类型（64位浮点数）以保证运算的精确性。而对于图像而言，这种数据类型在图像尺寸较大时可能并不理想。例如，一张1 000像素见方的图像拥有一百万个像素，如果每个像素用64位二进制数表示，总共需要大约8MB的内存空间。

为了减小图像信息的空间开销，可以将图像信息存为8位无符号整型数（uint8）和16位无符号整型数（uint16）的数组，这样只需要双精度浮点数八分之一或四分之一的空间。在上述3种存储类型中以双精度和uint8使用最多，uint16的情况与uint8大致类似。

2.3 Matlab的图像转换

1．图像存储格式的互相转换

有时必须将图像存储格式加以转换才能使用某些图像处理函数。例如，当使用某些Matlab内置的滤镜时，需要将索引图像转换为RGB图像或者灰度图像，Matlab才会将图像滤镜应用于图像数据本身，而不是索引图像中的颜色索引值表（这将产生无意义的结果）。

Matlab 提供了一系列存储格式转换函数，如表 2.11 所示。它们的名字都很便于记忆，例如ind2gray可以将索引图像转化为灰度图像。

表2.11 图像格式转换函数

函　数	描　述
dither	使用抖动的方式创建较小颜色信息量的图像。例如从灰度图转换成黑白图，或者从 RGB 图转换成索引图。多数时候返回 uint8 类型的图像，如果输出图像是包含大于 256 色颜色表的索引图，则使用 uint16 类型

续表

函　数	描　述
gray2ind	从灰度图转换成索引图。多数时候返回 uint8 类型的图像，如果输出图像是包含大于 256 色颜色表的索引图，则使用 uint16 类型。 [X,MAP] = gray2ind(*I*,*N*)，输出 *X* 为图像数据，MAP 为颜色表；输入 *I* 为原始图像，*N* 为索引颜色数目
grayslice	使用阈值法从灰度图创建索引图。多数时候返回 uint8 类型的图像，如果输出图像是包含大于 256 色颜色表的索引图，则使用 uint16 类型。 *X*=grayslice(*I*,*N*) *X*=grayslice(*I*,*V*) *X* 为输出的索引图像，*N* 为需要均匀划分的阈值个数，*V* 为给定的阈值向量
im2bw	使用阈值法从灰度图、索引图或 RGB 图创建二值图。返回逻辑型矩阵存储的图像。 BW = im2bw(I,LEVEL)或 BW = im2bw(X,MAP,LEVEL)，LEVEL 为指定的阈值。关于计算与确定阈值的方法，在后面的章节中介绍
ind2gray	从索引图创建灰度图。返回图像与原图像存储类型相同。 *I* = ind2gray(*X*,MAP)
ind2rgb	从索引图创建 RGB 图，返回 double 类型存储的图像。 RGB = ind2rgb(*X*,MAP)
mat2gray	使用归一化方法将一个矩阵中的数据扩展成对应的灰度图，返回图像使用 double 类型存储。 *I* = mat2gray(*A*,[AMIN AMAX])，AMIN 和 AMAX 指定了函数在转换时使用的下限和上限。*A* 中低于 AMIN 和高于 AMAX 的数据将被截取到 0 和 1
rgb2gray	从 RGB 图创建灰度图，返回图像与原图像存储类型相同。 *I* = rgb2gray(RGB) 也可以处理颜色表，调用方式如下。 NEWMAP = rgb2gray(MAP)，此时输入输出类型均为 double
rgb2ind	从 RGB 图创建索引图。多数时候返回 uint8 类型的图像，如果输出图像是包含大于 256 色颜色表的索引图，则使用 uint16 类型。 [X,MAP] = rgb2ind(RGB,N) X = rgb2ind(RGB,MAP) *N* 为颜色表中颜色的数目，MAP 为输出或给定的颜色表

也可以使用一些矩阵操作函数实现某些格式转换。例如，下面的语句可以将一幅灰度图像转换为RGB图像。

RGBIMAGE = CAT(3, GRAY, GRAY, GRAY);

2．图像数据类型转换

Matlab图像处理工具箱中支持的默认图像数据类型是uint8，使用imread函数读图像文件一般都为uint8类型。然而，很多数学函数如sin等并不支持double以外的类型，例如当试图对 uint8类型直接使用sin函数进行操作时，Matlab会提示如下的错误信息。

I = imread(‘coins.png’); %读入一幅unit8图像

sin(I);

??? Undefined function or method 'sin' for input arguments of type 'uint8'

针对这种情况，除了使用2.1.4小节介绍的强制类型转换方法外，还可利用图像处理工具箱中的内置图像数据类型转换函数。内置转换函数的优势在于它们可以帮助处理数据偏移量和归一化变换，从而简化了我们的编程工作。

一些常用的图像类型转换函数如表2.12所示。

表2.12 图像数据类型转换函数

im2uint8	将图像转换为 uint8 类型
im2uint16	将图像转换为 uint16 类型
im2double	将图像转换为 double 类型

可以在使用Matlab数学函数前将图像转换为double类型，而在准备将图像写入文件时再将其转换为uint8类型，如下所示。

I_d = im2double(I_uint8); %将uint8图像转换为double类型，灰度范围也相应从[0，255]归一化至[0，

1]

Iout_d = sin(I_d); %进行数学计算

Iout_uint8= im2uint8(Iout_d); %转换回uint8（灰度范围也重新扩展到[0，255]）, 准备写入文件

2.4 读取和写入图像文件

Matlab可以处理以下的图像文件类型：BMP、HDF、JPEG、PCX、TIFF、XWD、ICO、GIF、CUR。可以使用imread和imwrite函数对图像文件进行读写操作，使用imfinfo函数来获得数字图像的相关信息。

1．imread函数

imread函数可以将指定位置的图像文件读入工作区。对于除索引图像以外的情况，其原型如下。

A = imread(FILENAME, FMT);

参数说明

● FILENAME 指定图像文件的完整路径和文件名。如果要读入的文件在当前工作目录中或者自动搜索列表中给出的路径下，则只需提供文件名。

● FMT 参数指定图像文件的格式所对应的标准扩展名，例如 GIF 等。如果 imread 没有找到FILENAME所指定的文件，它会尝试FILENAME.FMT。

返回值

● A是一个包含图像数据的矩阵。对于灰度图，它是一个m行n列的矩阵；对于RGB真彩图，则是一个m×n×3的矩阵。对于大多数图像文件，A的类型为uint8；而对于某些TIFF和PNG图像，A的类型为uint16。

对于索引图像，情况有所不同，此时IMREAD的调用形式如下。

[X, MAP] = imread(FILENAME, FMT);

此时的返回值中，X 为图像数据矩阵，MAP 则是颜色索引表。图像中的颜色索引数据会被归一化到0到1的范围内。因为，对于索引图像，不论图像文件本身使用何种数据类型，imread函数都会使用双精度类型存储图像数据。

imread函数还可以处理RGBA等格式存储的图像，可以通过在命令窗口中输入help imread来查看Matlab中有关imread的在线帮助信息。

2．IMWRITE函数

imwrite 将指定的图像数据写入文件中，通过指定不同的保存文件扩展名，可以起到图像格式转换的作用（参见例2.4）。其调用格式如下。

imwrite(A, FILENAME, FMT);

● FILENAME参数指定文件名（不必包含扩展名）。

● FMT参数指定保存文件所采用的格式。

存储索引图像时，还需要一并存储颜色索引表，则此时IMWRITE函数的使用方法应为如下所示。

imwrite(A, MAP, FILENAME, FMT);

MAP是合法的Matlab颜色索引表。

imwrite函数还可以控制图像文件的很多属性，如TIFF文件格式所选择的彩色空间、GIF格式中的透明色以及图像文件的作者、版权信息、解析度和创建软件等。

[例2.4]读入一幅tif图像文件，并在写入磁盘时将tif图像转换为bmp图像。

```
>>I=imread('pout.tif'); %读入图像
>>whos I %查看图像变量信息
Name     Size         Bytes    Class
I        291x240      69840    uint8 array
Grand total is 69840 elements using 69840 bytes
```

%通过whos命令可以看到读入的高为291、宽为240的灰度图像I就是一个291*240的二维矩阵

>>imwrite(I, 'pout.bmp'); %将图像写入文件pout.bmp，同时起到了转换文件类型的作用

例2.4中的功能已经被封装在金羽图书论坛（http://bbs.book95.com）的“金羽图书与答疑”板块与本书同名的主题帖子附件中的“Chapter2/Code目录下的ex2_4.m”文件内。

3．imfinfo函数

imfinfo函数可以读取图像文件中的某些属性信息，例如修改日期、大小、格式、高度、宽度、色深、颜色空间、存储方式等。

imfinfo(FILENAME, FMT);

● FILENAME参数指定文件名。

● FMT参数是可选参数，指定文件格式。

[例2.5]查看图像文件信息。

```
>>imfinfo('pout.tjf') %查看图像文件信息
ans =
                  Filename: 'F:\Program
Files\Matlab\R2011a\toolbox\images\imdemos\pout.
tif'
               FileModDate: '04-十二月-2000 13:57:50'
                  FileSize: 69004
                    Format: 'tif'
             FormatVersion: []
                     Width: 240
                    Height: 291
                  BitDepth: 8
                 ColorType: 'grayscale'
           FormatSignature: [73 73 42 0]
                 ByteOrder: 'little-endian'
            NewSubFileType: 0
             BitsPerSample: 8
               Compression: 'PackBits'
 PhotometricInterpretation: 'BlackIsZero'
              StripOffsets: [9x1 double]
           SamplesPerPixel: 1
              RowsPerStrip: 34
           StripByteCounts: [9x1 double]
```

```
          XResolution: 72
          YResolution: 72
       ResolutionUnit: 'None'
             Colormap: []
  PlanarConfiguration: 'Chunky'
            TileWidth: []
           TileLength: []
          TileOffsets: []
       TileByteCounts: []
          Orientation: 1
            FillOrder: 1
     GrayResponseUnit: 0.0100
       MaxSampleValue: 255
       MinSampleValue: 0
         Thresholding: 1
               Offset: 68754
```

2.5 图像的显示

Matlab中一般使用imshow函数来显示图像，该函数可以创建一个图像对象，并可以自动设置图像的诸多属性，从而简化编程操作。这里介绍imshow函数的几种常见调用方式。

1．imshow函数

imshow 函数用于显示工作区或图像文件中的图像，在显示的同时可控制部分效果，常用的调用形式如下。

imshow(I, [low high], param1, value1, param2, value2, …)

imshow(I, MAP)

imshow(filename)

●I为要显示的图像矩阵。

● 可选参数[low high]指定显示灰度图像时的灰度范围，灰度值低于low的像素被显示为黑色，高于high的像素被显示为白色，介于low和high之间的像素被按比例显示为各种等级的灰色。如果将此参数指定为空矩阵[ ]，则函数会将图像矩阵中的最小值指定为low，最大值指定为high，从而达到灰度拉伸的显示效果。这个参数常常用于改善灰度图像的显示效果。

● 可选参数param1、value1、param2、value2等可以用来指定显示图像的特定方法。

● MAP为颜色索引表，除了显示索引图像，这在显示伪彩色图像时也可用到。

● filename参数指定图像文件名，这样可以不必将图像文件首先读入工作区。

[例2.6]图像文件的读取、显示、回写。

ex2_6.m

```
% 读取图像文件
>>I = imread('gantrycrane.png');
% 显示图像
>>imshow(I);
% 写回到文件
>>imwrite(I, 'gantrycrane.tif', 'TIFF');
```

2．多幅图像的显示

有时需要将多幅图像一起显示以比较它们之间的异同，这在考察不同算法对同一幅图像的处理效果时尤为有用。

可以在同一窗口或者不同的窗口显示多幅图像，这两种方式的实现见例2.7。

[例2.7]显示多幅图像。

```
% ex2_7.m
I = imread('pout.tif'); % 读取图像
% 在不同窗口显示
figure; %创建一个新的窗口
imshow(I);
figure;
imshow(I, [ ]);
% 在相同窗口显示
figure;
subplot(1, 2,1);
imshow(I);
subplot(1,2,2);
imshow(I, [ ]);
```

上述程序中，figure 函数用于新创建一个显示窗口，从而避免新图像的显示覆盖原图像；subplot(m,n,p)函数的含义是打开一个有m行n列图像位置的窗口，并将焦点位于第p个位置上。

注意

在多幅索引图像的显示中存在着潜在的问题。由于索引图像使用的颜色表可能不同，而系统的全局颜色表在默认情况下是8位的，最多只能存储256种颜色，这样一来，如果所有图像总颜色种类超过256种，则超出的部分将不会被正确显示。所以，通常先使用 ind2rgb(I)将图像转换为 RGB 模式。此外，也可以使用subimage(I,map)，这个函数在显示图像之前会自动将其转换为RGB格式。

3．多帧图像的显示

在显示多帧图像时，可以显示多帧中的一帧，或者将它们显示在同一个窗口内，也可以将多帧图像转化成电影播放出来。这3种方式的实

现见例2.8。

[例2.8]多幅图像D，存储了一组索引图像，MAP为颜色索引表。分别以上述的3种方式显示它们。

```
>>load mri    %载入Matlab自带的核磁共振图像
>>imshow(D(:,:,7), map);    % 显示多幅中的一幅
% 同一窗口显示
>>figure, montage(D, map);
% 转化成为电影
>>figure
>>mov=immovie(D, map);
>>colormap(map);    %设定颜色表
>>movie(mov);    %播放电影
```

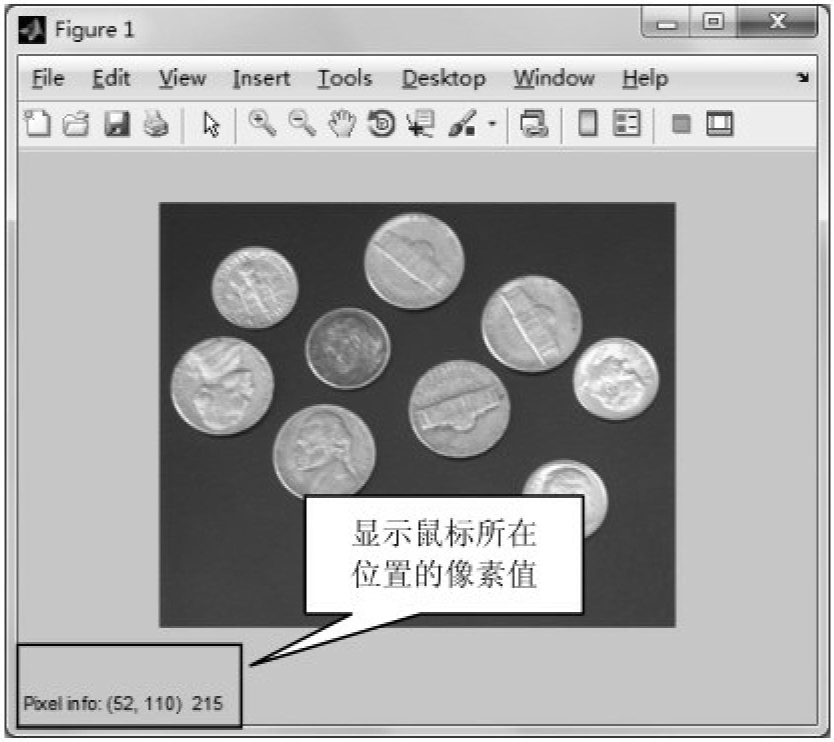
金羽图书论坛（http://bbs.book95.com）的“金羽图书与答疑”板块与本书同名的主题帖子附件中“chapter2/code”目录下的ex2_8.m文件封装了例2.8中的功能。

4．图像的放缩

有时需要将图像的某一部分放大以查看局部的详细情况。只需输入zoom on命令即可实现图像的放缩，zoom off可以关闭图像放缩功能。打开图像放缩功能之后，就可以通过简单的鼠标操作观察图像细部了。

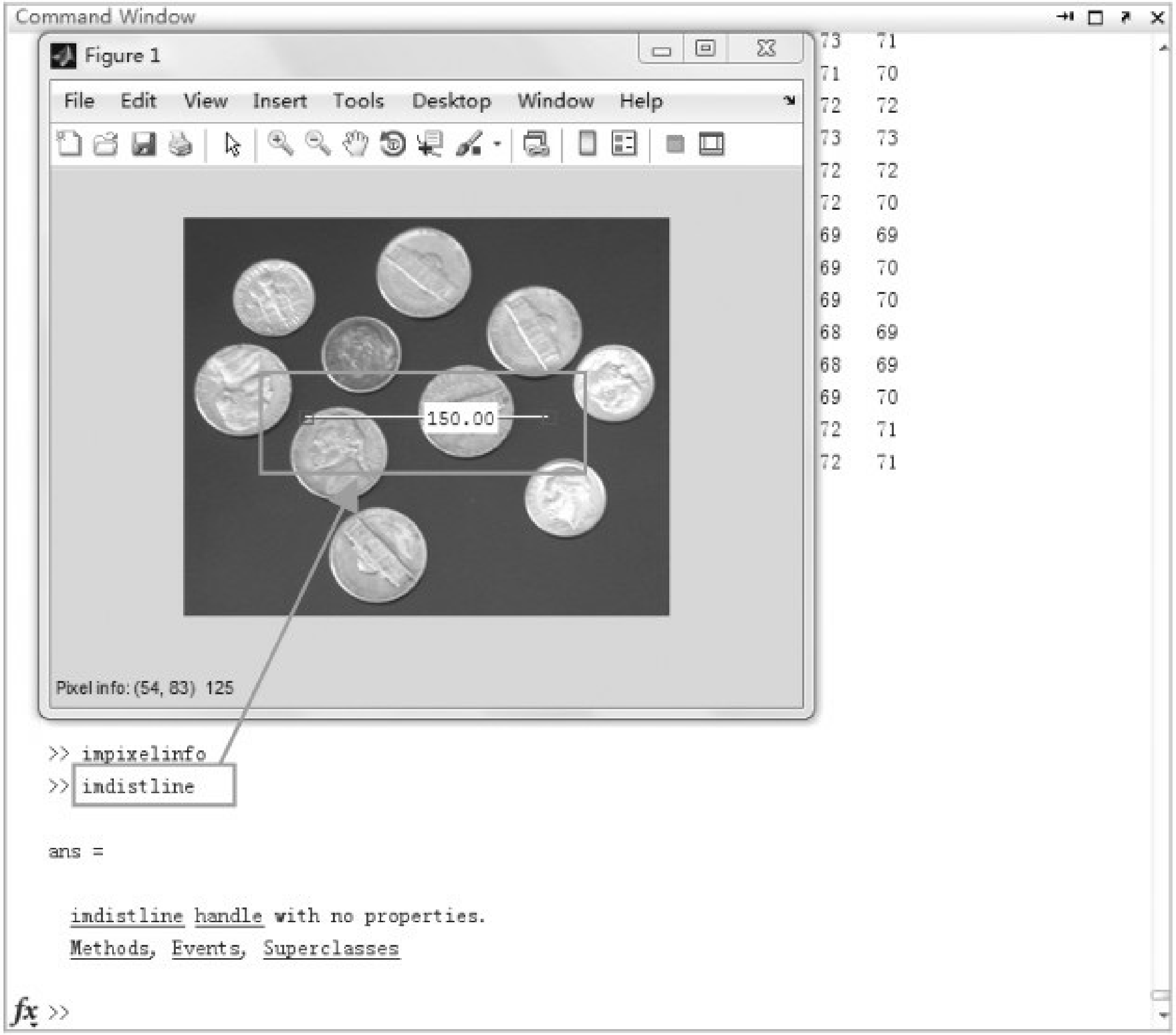
5．像素值查看工具

使用imshow函数显示一幅图像之后，可以通过输入impixelinfo命令在最后显示的图像窗口的左下角，随鼠标光标的移动显示鼠标指针所指位置处的像素值，如图2.12所示。



▲图2.12 像素查看器

还可以通过imdistline命令以交互的方式查看图像中两点之间的距离，如图2.13所示。



▲图2.13 使用 imdistline命令查看距离

第3章 图像的点运算

对于一个数字图像处理系统来说，一般可以将处理流程分为3个阶段。在获取原始图像后，首先是图像预处理阶段，第二是特征抽取阶段，最后才是识别分析阶段。预处理阶段尤为重要，这个阶段处理不好则后面的工作根本无法展开。

点运算指的是对图像中的每个像素依次进行同样的灰度变换运算。设r和s分别是输入图像f(x,y)和输出图像g(x,y)在任一点(x,y)的灰度值，则点运算可以使用下式定义。

$$s = T(r) \tag{3-1}$$

其中，T为采用的点运算算子，表示在原始图像和输出图像之间的某种灰度级映射关系。

点运算常常用于改变图像的灰度范围及分布，是图像数字化及图像显示时常常需要的工具。点运算因其作用的性质有时也被称为对比度①增强、对比度拉伸或灰度变换。

【注】①对比度：灰度图像最大亮度与最小亮度的比值。

本章的知识和技术热点

- 最基本的图像分析工具——灰度直方图。
- 利用直方图辅助实现的各种灰度变换，包括灰度线性变换、伽马变换、阈值变换和窗口变换等。
- 两种实用的直方图修正技术——直方图均衡化和直方图规定化。

本章的典型案例分析

- 基于直方图均衡化的图像灰度归一化。

● 直方图规定化。

3.1 灰度直方图

灰度直方图描述了一幅图像的灰度级统计信息，主要应用于图像分割和图像灰度变换等处理过程中。

3.1.1 理论基础

从数学上来说，图像直方图描述图像的各个灰度级的统计特性，它是图像灰度值的函数，统计一幅图像中各个灰度级出现的次数或概率。有一种特殊的直方图叫做归一化直方图，可以直接反映不同灰度级出现的比率。

从图形上来说，灰度直方图是一个二维图，横坐标为图像中各个像素点的灰度级别，纵坐标表示具有各个灰度级别的像素在图像中出现的次数或概率。

提示

如不特别说明，本书中的直方图的纵坐标都对应着该灰度级别在图像中出现的次数，而归一化直方图的纵坐标则对应着该灰度级别在图像中出现的概率。

灰度直方图的计算是根据其统计定义进行的。图像的灰度直方图是一个离散函数，它表示图像每一灰度级与该灰度级出现频率的对应关系。假设一幅图像的像素总数为N，灰度级总数为L，其中灰度级为g的像素总数为N_g，则这幅数字图像的灰度直方图横坐标即为灰度g（$0 \le g \le L-1$），纵坐标则为灰度值出现的次数N_g。实际上，用像素总数N去除以各个灰度值出现的次数N_g即得到各个灰度级出现的概率$P_g=N_g/N=N_g/\Sigma N_g$，从而得到归一化的灰度直方图，其纵坐标为概率P_g。

3.1.2 Matlab实现

Matlab中的imhist函数可以进行图像的灰度直方图运算，调用语法如下。

```
imhist(I)
imhist(I, n)
[counts,x] = imhist(...)
```

参数说明

- I为需要计算灰度直方图的图像。
- n为指定的灰度级数目。如果指定参数n，则会将所有的灰度级均匀分布在n个小区间内，而不是将所有的灰度级全部分开。

返回值

- counts为直方图数据向量。counts(i)表示第i个灰度区间中的像素数目。
- x是保存了对应的灰度小区间的向量。

若调用时不接收这个函数的返回值，则直接显示直方图；在得到这些返回数据之后，也可以使用stem(x, counts)来手工绘制直方图。

1．一般直方图

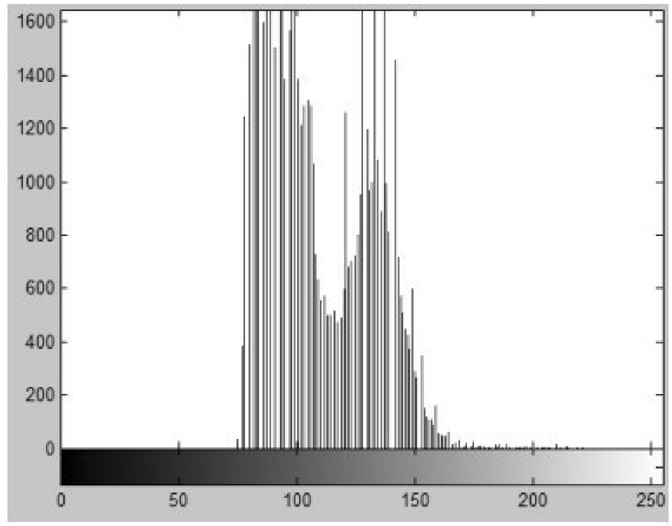
下面使用了Matlab中的一张内置示例图片演示灰度直方图的生成与显示，程序如下。

```
I = imread('pout.tif');    %读取图像pout.tif
figure;    %打开一个新窗口
imshow(I); title('Source');    %显示原图像
figure;    %打开一个新窗口
imhist(I); title('Histogram');    %显示直方图
```

上述程序的运行结果如图3.1所示。



（a）示例图片 pout.tif



（b）灰度直方图

▲图3.1 灰度直方图的生成

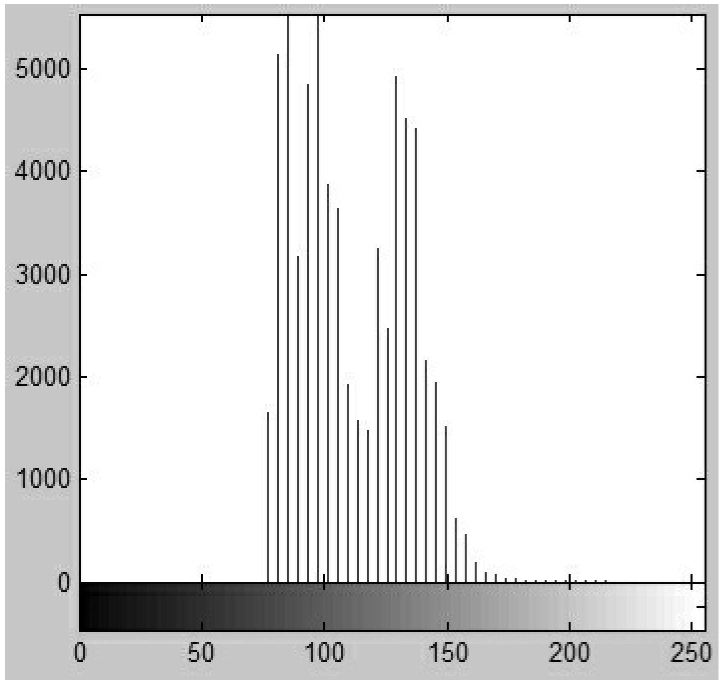
图3.1（b）中未经归一化的灰度直方图的纵轴表示图像中所有像素取到某一特定灰度值的次数；横轴对应从0～255的所有灰度值，覆盖了uint8存储格式的灰度图像中的所有可能取值。

因为相近的灰度值所具有的含义往往是相似的，所以常常没有必要在每个灰度级上都进行统计。下面的命令将0～255总共256个灰度级平均划分为64个长度为4的灰度区间，此时纵轴分别统计每个灰度区间中的像素在图像中的出现次数。

```
imhist(I, 64);    % 生成有64个小区间的灰度直方图
```

上述命令执行后效果如图3.2所示。

由于要统计落入每个灰度区间内的像素数目，灰度区间常常被形象地称为“收集箱”。在图3.2所示的直方图中，由于减少了收集箱的数目，使得落入每个收集箱的像素数目有所增加，从而使直方图更具统计特性。收集箱的数目一般设为2的整数次幂，以保证可以无需圆整。



▲图3.2 分为 64段小区间的灰度直方图

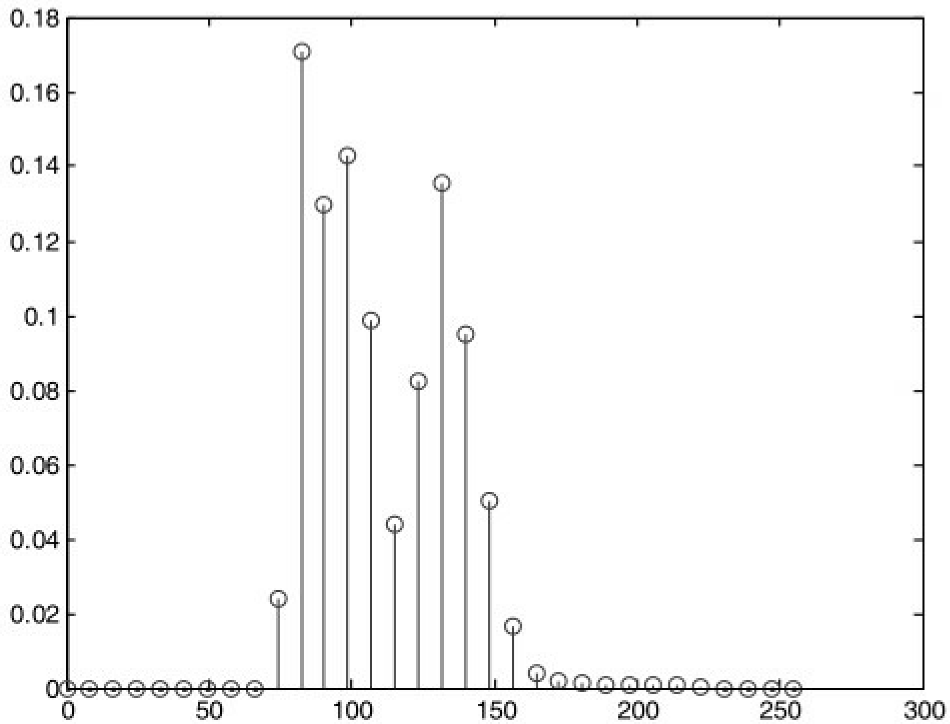
2．归一化直方图

在imhist函数的返回值中，counts保存了落入每个区间的像素的个数，通过计算counts与图像中像素总数的商可以得到归一化的直方图。

绘制有32个灰度区间的归一化直方图的Matlab程序如下。

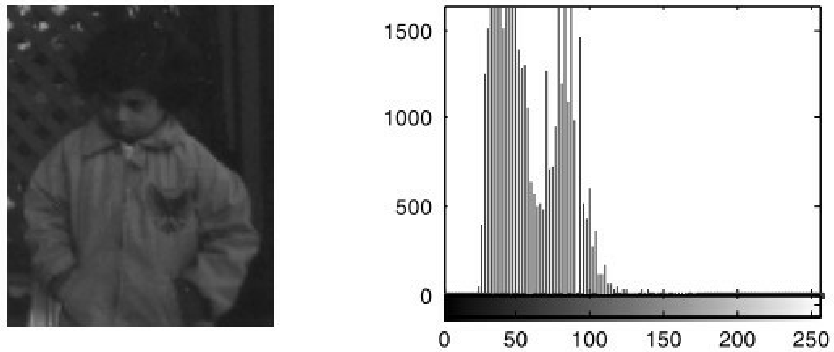
```
I = imread('pout.tif');    %读入原图像
figure;    %打开新窗口
[M,N] = size(I);    %计算图像大小
[counts, x] = imhist(I, 32);    %计算有32个小区间的灰度直方图
counts = counts / M / N;    %计算归一化灰度直方图各区间的值
stem(x, counts);    %绘制归一化直方图
```

上述程序的运行的结果如图3.3所示。

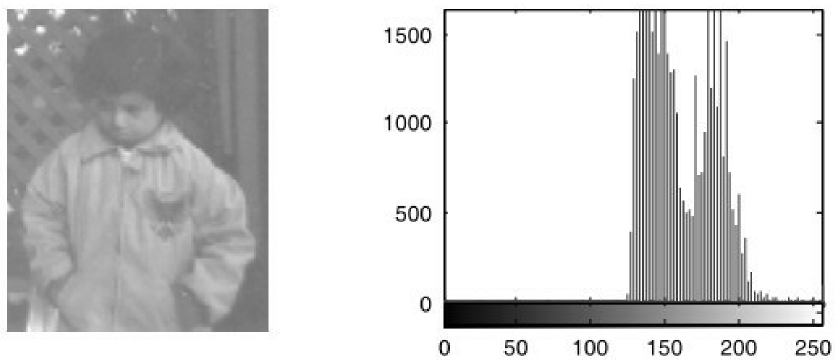


▲图3.3 归一化直方图

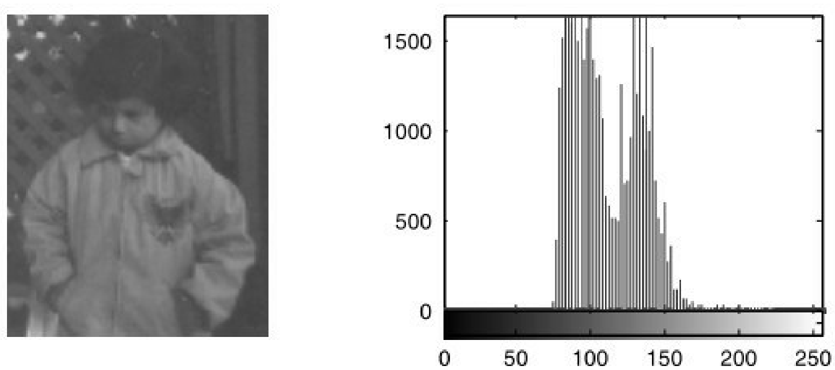
分析图像的灰度直方图往往可以得到很多有效的信息。例如，从图3.4所示的一系列灰度直方图上，可以很直观地看出图像的亮度和对比度特征。实际上，直方图的峰值位置说明了图像总体上的亮暗：如果图像较亮，则直方图的峰值出现在直方图的较右部分；如果图像较暗，则直方图的峰值出现在直方图的较左部分，从而造成暗部细节难以分辨。如果直方图中只有中间某一小段非零值，则这张图像的对比度较低；反之，如果直方图的非零值分布很宽而且比较均匀，则图像的对比度较高。



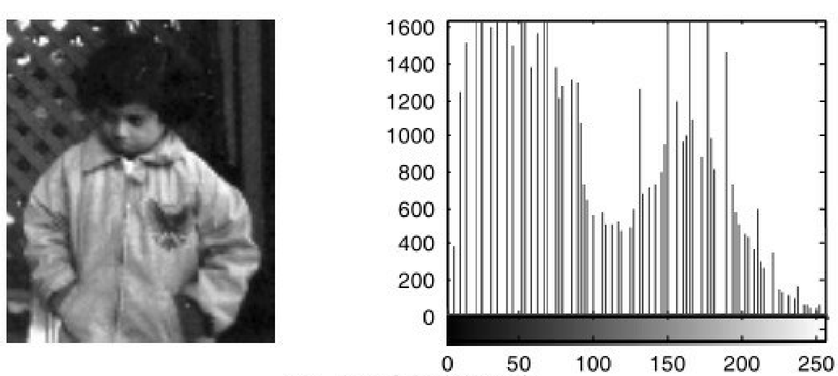
（a）较暗的图像



（b）较亮的图像



（c）对比度较小的图像



（d）对比度较大的图像

▲图3.4 图像的灰度直方图与其亮度、对比度的关系

上面列举的情况，均可以通过以直方图为依据的图像增强方法进行处理，这些方法的具体介绍详见3.7节。

3.2 灰度的线性变换

灰度线性变换是最常用的图像点运算之一，将图像的像素值通过指定的线性函数进行变换，以此增强或减弱图像的灰度。

3.2.1 理论基础

线性灰度变换函数f(x)是一个一维线性函数。

$$D_B = f(D_A) = f_A D_A + f_B \qquad (3\text{-}2)$$

式中：参数f_A为线性函数的斜率；f_B为线性函数在y轴的截距；D_A表示输入图像的灰度；D_B表示输出图像的灰度。

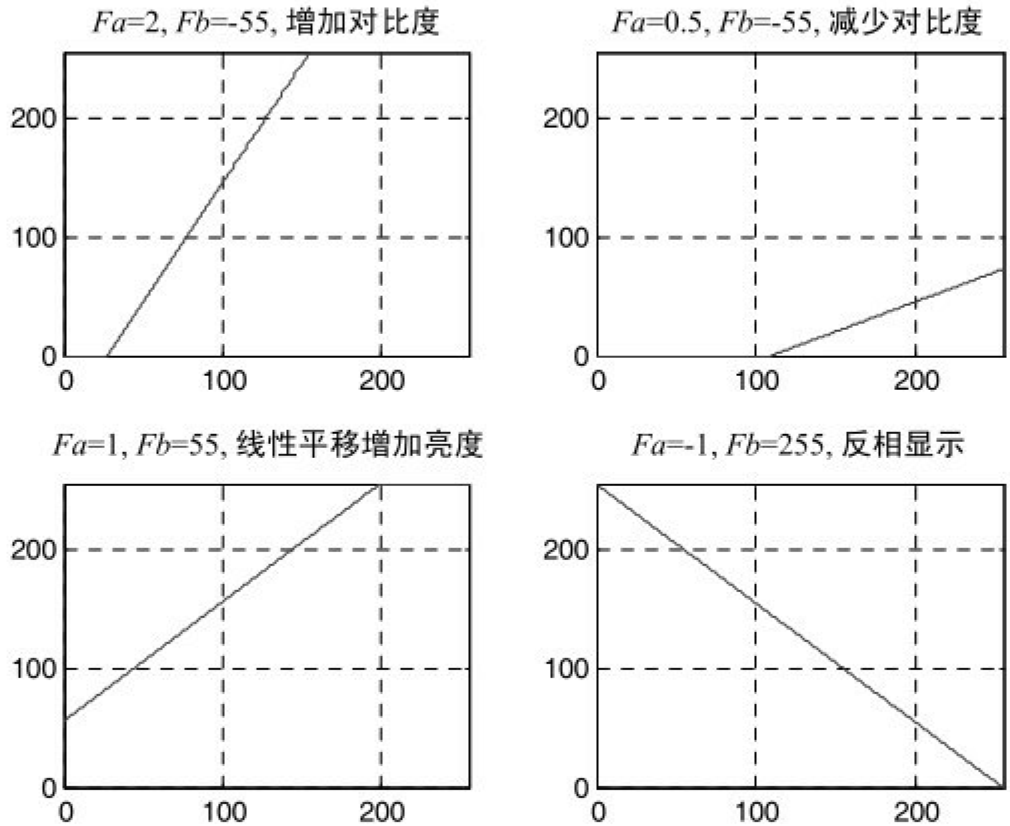
● 当$f_A>1$时，输出图像的对比度将增大；当$f_A<1$时，输出图像的对比度将减小。

● 当$f_A=1$且$f_B\neq0$时，操作仅使所有的像素的灰度值上移或下移，其效果是使整个图像更暗或更亮；如$f_A<0$，暗区域将变亮，亮区域将变暗。这种线性改变亮度的变换可能由于像素亮度达到饱和（小于0或超过255）从而丢失一部分细节。

● 特殊情况下，当 $f_A=1$，$f_B=0$时，输出图像与输入图像相同；当$f_A= -1$，$f_B=255$时，输出图像的灰度正好反转。灰度反转处理适用于增强暗色图像中的亮度较大的细节部分，这也是由人的视觉特性决定的。

图 3.5 给出了一些对应于上述情况的变换实例，后面的Matlab实现

中将分别展示应用这些变换的实际效果。



▲图3.5 线性变换示意图

3.2.2 Matlab实现

使用Matlab对图像执行线性变换无需专门的函数，下面的例3.1对Matlab示例图像coins.jpg进行了不同参数的线性变换操作。

[例3.1]不同参数的线性变换效果

本例中对Matlab示例图像coins.png进行了如图3.5中所示的增加对比度、减小对比度、线性平移增加亮度和反相显示4种线性变换，给出了变换效果以及其对应的直方图变化情况。

ex3_1.m

```
I = imread('coins.png');     %读入原图像
```

```
I = im2double(I);                  %转换数据类型为double
[M,N] = size(I);                   %计算图像面积
figure(1);                         %打开新窗口
imshow(I);                         %显示原图像
title('原图像');
figure(2);                         %打开新窗口
[H,x] = imhist(I, 64);        %计算64个小区间的灰度直方图
stem(x, (H/M/N), '.');        %显示原图像的直方图
title('原图像');
% 增加对比度
Fa = 2; Fb = -55;
O = Fa .* I + Fb/255;
figure(3);
subplot(2,2,1);
imshow(O);
title('Fa = 2 Fb = -55 增加对比度');
figure(4);
subplot(2,2,1);
[H,x] = imhist(O, 64);
stem(x, (H/M/N), '.');
title('Fa = 2 Fb = -55 增加对比度');
% 减小对比度
Fa = 0.5; Fb = -55;
O = Fa .* I + Fb/255;
figure(3);
subplot(2,2,2);
imshow(O);
```

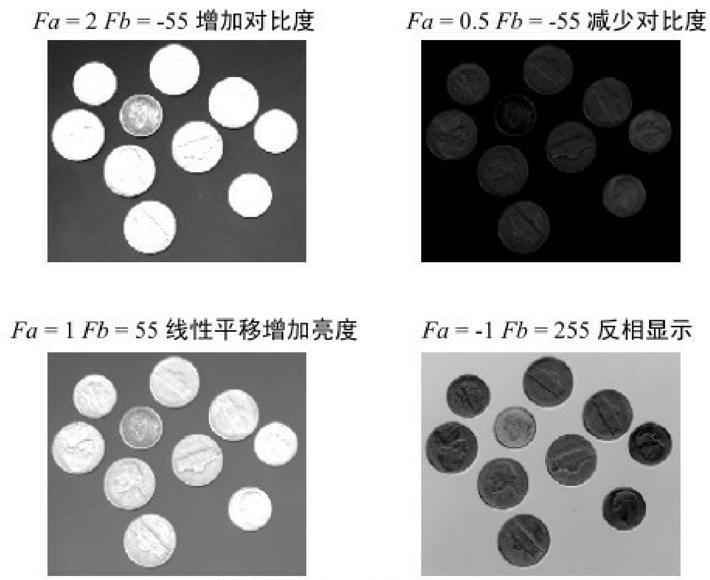
```
title('Fa = 0.5 Fb = -55 减小对比度');
figure(4);
subplot(2,2,2);
[H,x] = imhist(O, 64);
stem(x, (H/M/N), '.');
title('Fa = 0.5 Fb = -55 减小对比度');
% 线性增加亮度
Fa = 1; Fb = 55;
O = Fa .* I + Fb/255;
figure(3);
subplot(2,2,3);
imshow(O);
title('Fa = 1 Fb = 55 线性平移增加亮度');
figure(4);
subplot(2,2,3);
[H,x] = imhist(O, 64);
stem(x, (H/M/N), '.');
title('Fa = 1 Fb = 55 线性平移增加亮度');
% 反相显示
Fa = -1; Fb = 255;
O = Fa .* I + Fb/255;
figure(3);
subplot(2,2,4);
imshow(O);
title('Fa = -1 Fb = 255 反相显示');
figure(4);
subplot(2,2,4);
```

```
[H,x] = imhist(O, 64);
stem(x, (H/M/N), '.');
title('Fa = -1 Fb = 255 反相显示');
```

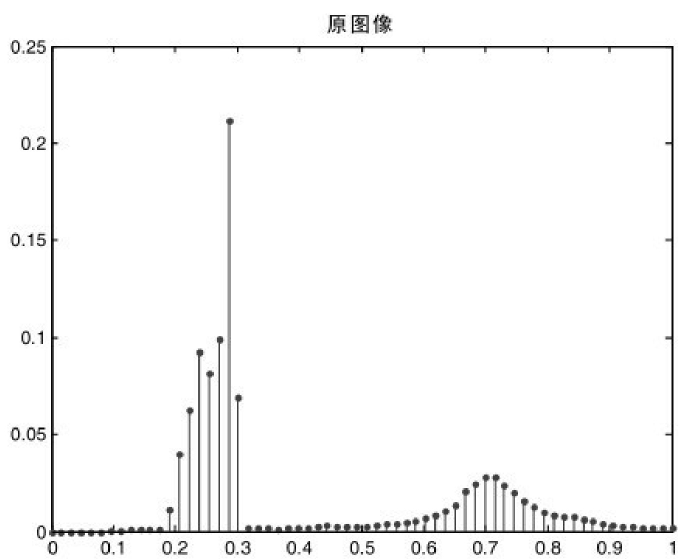
上述程序存放在金羽图书论坛（http://bbs.book95.com）的“金羽图书与答疑”板块与本书同名的主题帖子附件中“Chapter3/Code”目录下的ex3_1.m文件中，其运行结果如图3.6所示。



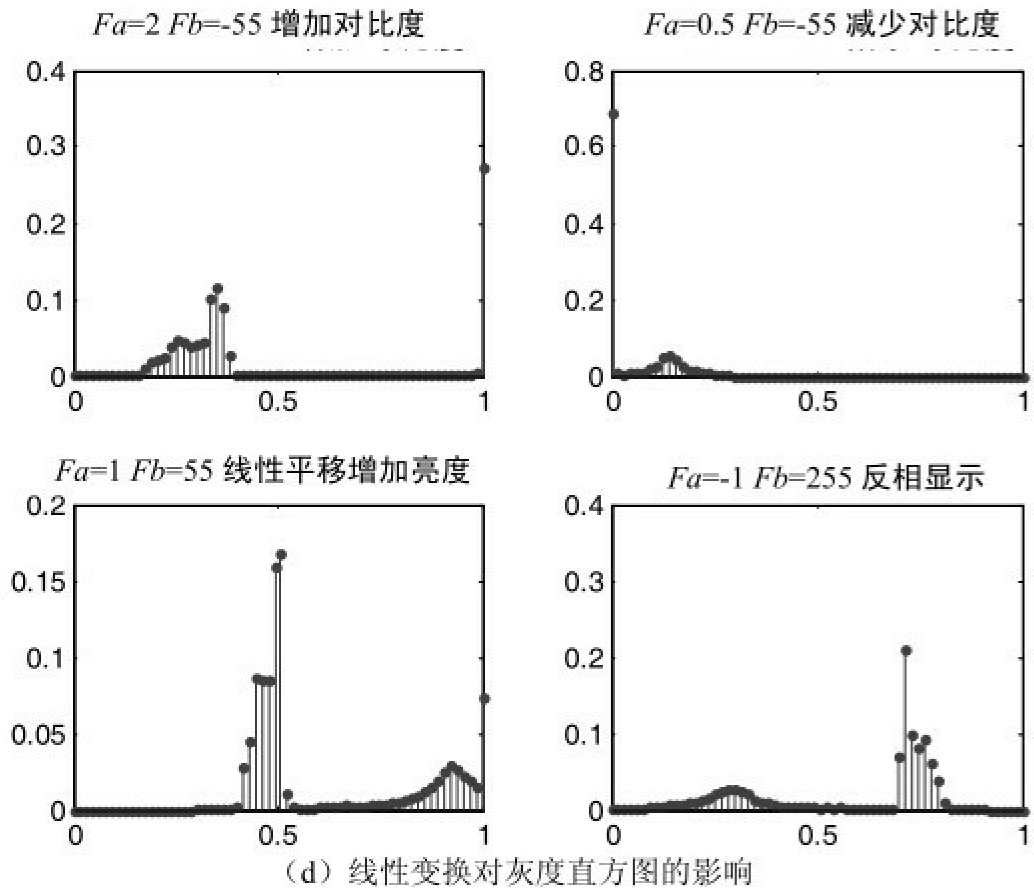
（a）示例图像 coins.png



（b）线性变换对图像的影响



（c）原图像的灰度直方图



▲图3.6 线性变换实例说明

从图 3.5、图 3.6 可以看出：改变图像的对比度是对直方图的缩放与平移，改变图像的亮度则只是平移直方图在横轴上的位置，而反相则是将直方图水平镜像。

单纯的线性灰度变换可以在一定程度上解决视觉上的图像整体对比度问题，但是对图像细节部分的增强则较为有限，结合后面将要介绍的非线性变换技术可以解决这一问题。

3.3 灰度对数变换

本节介绍一种灰度的非线性变换——对数变换，并介绍它在傅立叶

频谱显示中的应用。

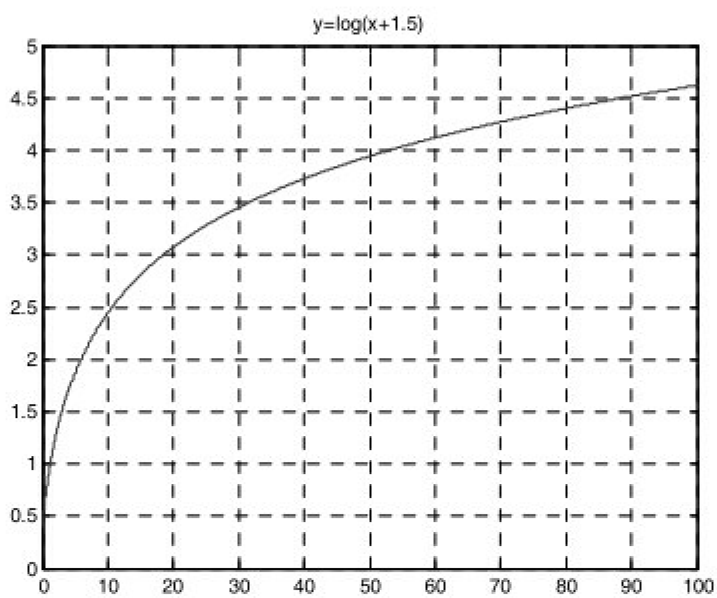
3.3.1 理论基础

对数变换的一般表达式如下。

$$t = c\ \log(1+s) \qquad (3\text{-}3)$$

其中c为尺度比例常数，s为源灰度值，t为变换后的目标灰度值。

在如图3.7所示的对数曲线上，函数自变量低值时，曲线的斜率很高；自变量高值时，曲线斜率变小。



▲图3.7 对数变换示意图

由对数函数曲线可知，这种变换可以增强一幅图像中较暗部分的细节，从而可用来扩展被压缩的高值图像中的较暗像素，因此对数变换被广泛地应用于频谱图像的显示中。一个典型的应用是傅立叶频谱（参见第 6 章），其动态范围可能宽达 $0 \sim 10^6$ 。直接显示频谱时，图像显示设备的动态范围往往不能满足要求，从而丢失了大量的暗部细节。而在

使用对数变换之后，图像的动态范围被合理地非线性压缩，从而可以清晰地显示。本节的Matlab实现中就提供了一个这样的示例。

3.3.2 Matlab实现

对数变换不需要专门的图像处理函数，可以使用如下数学函数实现对图像I的对数变换。

```
T = log(I + 1);
```

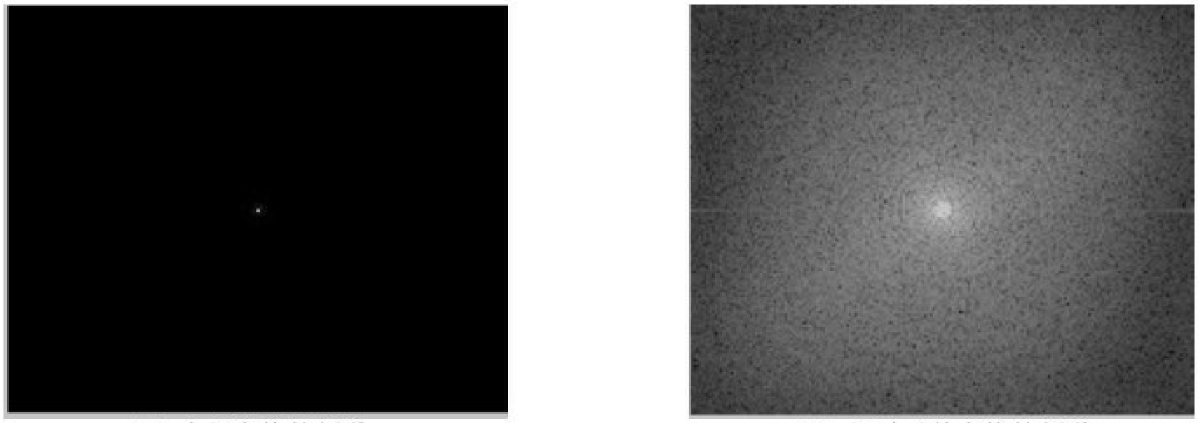
注意

log函数会对输入图像矩阵I中的每个元素进行操作，但是却仅能处理double类型的矩阵。从图像文件中得到的图像矩阵则大多数是uint8类型的，因此需要首先使用im2double函数来执行数据类型的转换。

下面的程序比较了对傅立叶频谱图像进行对数变换前后的效果（不必关注代码中生成傅立叶频谱的部分）。

```
I=imread('coins.png');   %读取图像
F = fft2(im2double(I));   %计算频谱
F = fftshift(F);
F = abs(F);
T$space@= log(F + 1);   %对数变换
subplot(1,2,1);
imshow(F, []);
title('未经变换的频谱');
subplot(1,2,2);
imshow(T, []);
title('对数变换后');   %显示原图和变换结果
```

上述程序的运行结果如图3.8所示。



（a）未经变换的频谱　　（b）经过对数变换的频谱

▲图3.8 对数变换效果示意图

由图3.8（a）中未经变换的频谱可见，图像中心绝对高灰度值的存在压缩了低灰度部分的动态范围，从而无法在显示时表现出细节；而经过对数灰度处理的图像，其低灰度区域对比度将会增加，暗部细节被增强。

3.4 伽玛变换

伽玛变换又称指数变换或幂次变换，是另一种常用的灰度非线性变换。

3.4.1 理论基础

伽玛变换的一般表达式如下。

$$y=(x+\mathrm{esp})^{\gamma} \quad (3\text{-}4)$$

式中：x与y的取值范围均为[0，1]；esp为补偿系数；γ则为伽玛系数。

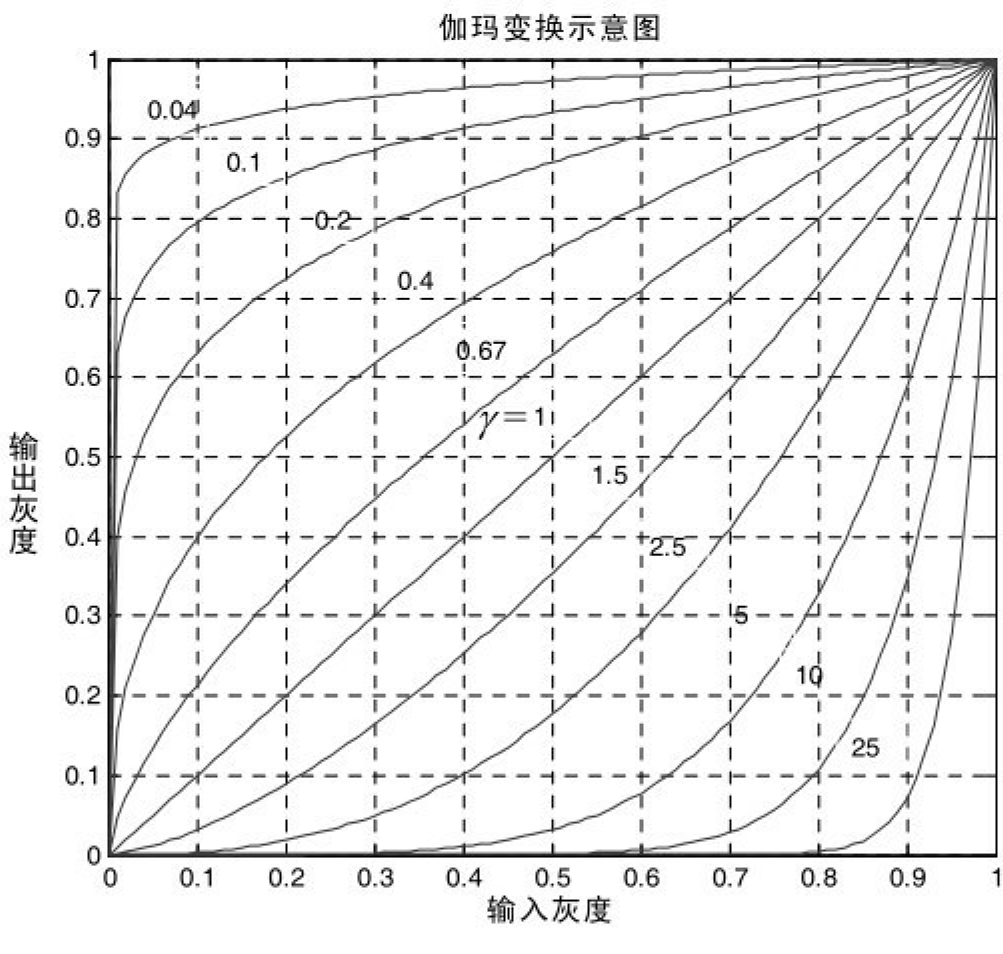
与对数变换不同，伽玛变换可以根据γ的不同取值选择性地增强低

灰度区域的对比度或是高灰度区域的对比度。

γ是图像灰度校正中非常重要的一个参数，其取值决定了输入图像和输出图像之间的灰度映射方式，即决定了是增强低灰度（阴影区域）还是增强高灰度（高亮区域）。

- γ>1时，图像的高灰度区域对比度得到增强。
- γ<1时，图像的低灰度区域对比度得到增强。
- γ=1时，这一灰度变换是线性的，即不改变原图像。

伽玛变换的映射关系如图3.9所示。在进行变换时，我们通常需要将0～255的灰度动态范围首先变换到0～1的动态范围，执行伽玛变换后再恢复原动态范围。



▲图3.9 0～1范围上的伽玛变换示意图

3.4.2 Matlab实现

Matlab中为我们提供了实现灰度变换的基本工具imadjust，它有着非常广泛的用途，其调用的一般语法如下。

J = imadjust(I, [low_in high_in], [low_out, high_out], gamma);

该函数将输入图像I中从low_in至high_in之间的值映射到输出图像J的low_out和high_out之间的值，low_in以下和high_in以上的值则被裁减掉。

参数说明

- [low_in high_in]和[low_out, high_out]确定源灰度范围到目标灰度范围的映射。在给定[low_in high_in]和[low_out, high_out]时，需要按照double类型给定，即取值范围在 0～1之间。

使[low_in high_in]和[low_out, high_out]为空（[ ]）相当于默认值[0 1]。若high_out小于 low_out，则输出图像J的亮度将会反转。

- 参数gamma指定了变换曲线的形状（类似于图3.9中的形状），其默认值为1，表示线性映射。若gamma<1，则映射被加权至更高的输出值；若gamma>1，则映射被加权至更低的输出值。

提示

当 gamma取 1时，通过设定合适的[low_in high_in]和[low_out high_out]的取值， imadjust 函数可以实现 3.2 节中的灰度线性变换；而当[low_in high_in]和[low_out high_out]的取值均为[0, 1]时，以不同的gamma调用 imadjust函数则可以实现图 3.9中所示的各种伽马变换。

- I为输入图像，可以是uint8、uint16或者double类型。

返回值

- J为经过处理的图像，与I具有同样的类型。

将imadjust函数用于Gamma变换的调用语法如下。

J = imadjust(I,[ ],[ ],gamma)

[例3.2]当gamma分别取不同值时的伽马变换效果。

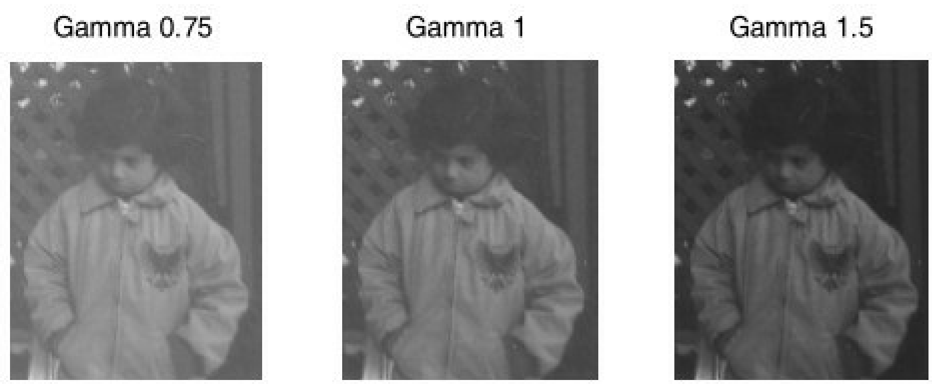
```
%ex3_2.m
I = imread(' pout.tif '); %读入原图像
% Gamma取0.75
subplot(1,3,1);
imshow(imadjust(I, [ ], [ ], 0.75));
title('Gamma 0.75');
% Gamma取1
subplot(1,3,2);
imshow(imadjust(I, [ ], [ ], 1));
title('Gamma 1');
% Gamma取1.5
subplot(1,3,3);
imshow(imadjust(I, [ ], [ ], 1.5));
title('Gamma 1.5');
```

上述程序的运行结果如图 3.10 所示，可以看出不同伽玛因子给图像的整体明暗程度带来的变化，以及对图像暗部和亮部细节清晰度的影响。当伽玛因子取1的时候，图像没有任何改变。

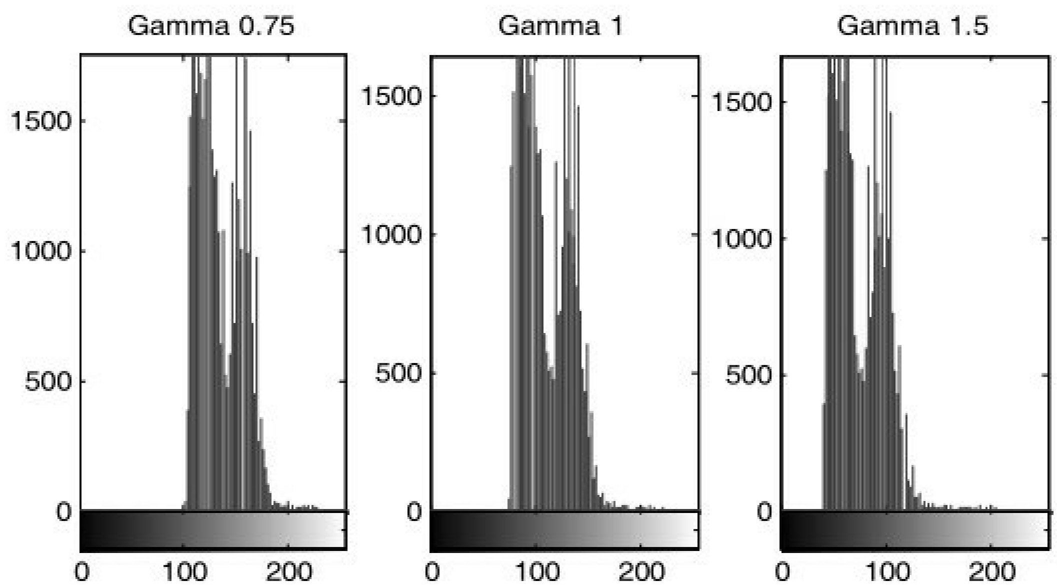
下面的程序生成了图3.11中3幅图像的灰度直方图。

```
% Gamma取0.75
subplot(1,3,1);
imhist(imadjust(I, [ ], [ ], 0.75));
title('Gamma 0.75');
% Gamma取1
subplot(1,3,2);
imhist(imadjust(I, [ ], [ ], 1));
title('Gamma 1');
```

```
% Gamma取1.5
subplot(1,3,3);
imhist(imadjust(I, [ ], [ ], 1.5));
title('Gamma 1.5');
```



▲图 3.10 伽玛变换效果



▲图3.11 伽玛变换对灰度直方图的影响

注意图 3.11 中直方图非零区间位置的变化，以及这些变化给图像带来的影响。由于伽玛变换并不是线性变换，所以它不仅可以改变图像的对比度，还能够增强细节，从而带来整体图像效果的增强和改善。

3.5 灰度阈值变换

灰度阈值变换可以将一幅灰度图像转换成黑白的二值图像。用户指定一个起到分界线作用的灰度值，如果图像中某像素的灰度值小于该灰度值，则将该像素的灰度值设置为0，否则设置为255。这个起到分界线作用的灰度值称为阈值，灰度的阈值变换也常被称为阈值化或二值化。

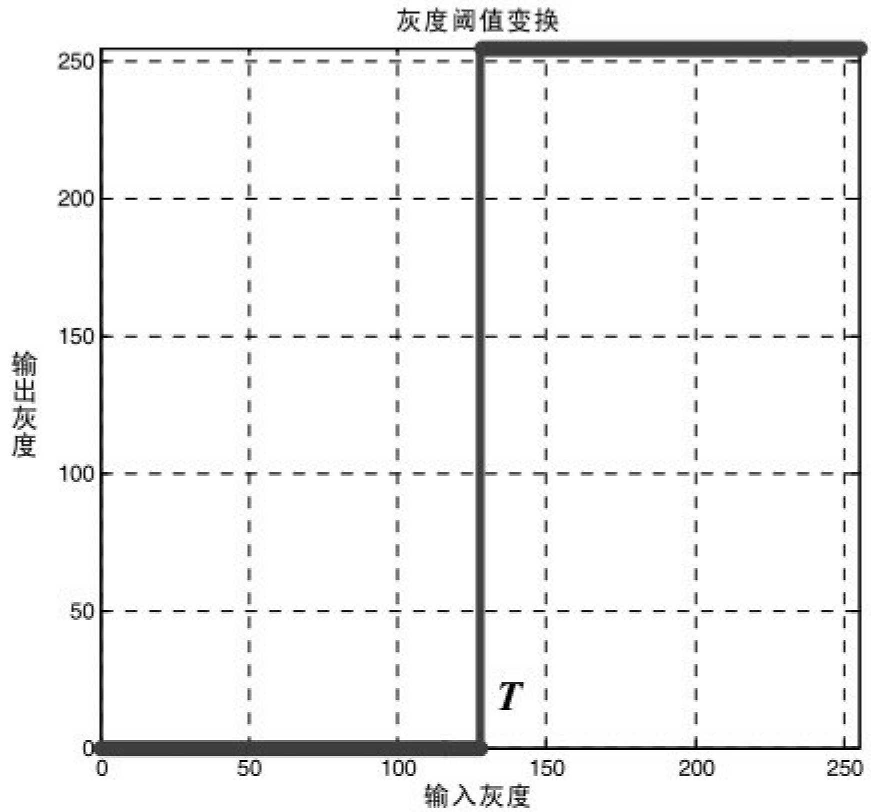
3.5.1 理论基础

灰度阈值变换的函数表达式如下。

$$f(x)=\begin{cases}0 & x<T \\ 255 & x\geqslant T\end{cases} \tag{3-5}$$

式中：T为指定的阈值。

图3.12给出了灰度阈值变换的示意图。



▲图3.12 灰度阈值变换示意图

灰度阈值变换的用途和可扩展性都非常广泛。通过将一幅灰度图像转为二值图像，可以将图像内容直接划分为我们关心的和不关心的两个部分，从而在复杂背景中直接提取出感兴趣的目标。因此它是图像分割的重要手段之一，这一点在第11章中还将进一步阐述。

3.5.2 Matlab实现

Matlab中和阈值变换有关的函数主要有两个——im2bw和graythresh，下面分别介绍。

1．函数**im2bw**可用于实现阈值变换，调用语法如下。

BW = im2bw(I, level)

参数说明

● 参数I为需要二值化的输入图像；

● 参数level给出了具体的变换阈值，它是一个0～1之间的双精度浮点数。例如输入图像I为灰度范围在0～255之间的uint8图像，如果level=0.5则对应于分割阈值128。

返回值

● BW为二值化后的图像。

2．函数**graythresh** 可以自适应地确定变换所用的**“**最优**”**阈值，调用形式如下。

thresh = graythresh(I)

参数说明

● 参数I为需要计算阈值的输入图像。

返回值

● thresh是计算得到的最优化阈值。

灰度阈值 level既可以由经验确定，也可以使用graythresh 函数来自适应地确定。下面的程序分别展示了如何利用graythresh函数获得的阈值和自行设定的阈值进行阈值变换。

```
>> I = imread('rice.png');    % 使用Matlab自带的 rice.png 图像
>> thresh = graythresh(I)    % 自适应确定阈值
thresh =
0.5137
>> bw1 = im2bw(I, thresh);    % 二值化
>>
>> bw2 = im2bw(I, 130/255);    % 以130为阈值实现二值化，注意要将此阈值转换至[0，1]区间
>> subplot(1,3,1);imshow(I);title('原图像');
>> subplot(1,3,2);imshow(bw1);title('自动选择阈值');
>> subplot(1,3,3);imshow(bw2);title('阈值130');
```

上述程序的运行结果如图3.13所示。



（a）原图像　（b）自动阈值　（c）阈值 130 手工设置

▲图3.13 灰度阈值变换效果

由3.13（b）和（c）可见，单纯的灰度阈值化无法很好地处理灰度变化较为复杂的图像，常常给物体的边缘带来误差，或者给整个画面带来噪点。这要通过其他的图像处理手段予以弥补，我们将在10.4节介绍相关的内容。

3.6 分段线性变换

分段线性变换有很多种，包括灰度拉伸、灰度窗口变换等，本节仅讲述最为常用的灰度拉伸。

3.6.1 理论基础

利用分段线性变换函数来增强图像对比度的方法实际是增强原图各部分的反差，即增强输入图像中感兴趣的灰度区域，相对抑制那些不感兴趣的灰度区域。分段线性函数的主要优势在于它的形式可任意合成，而其缺点是需要更多的用户输入。

分段线性变换的函数形式如下。

$$f(x)=\begin{cases}\dfrac{y_1}{x_1}x & x<x_1\\ \dfrac{y_2-y_1}{x_2-x_1}(x-x_1)+y_1 & x_1\leqslant x\leqslant x_2\\ \dfrac{255-y_2}{255-x_2}(x-x_2)+y_2 & x>x_2\end{cases} \tag{3-6}$$

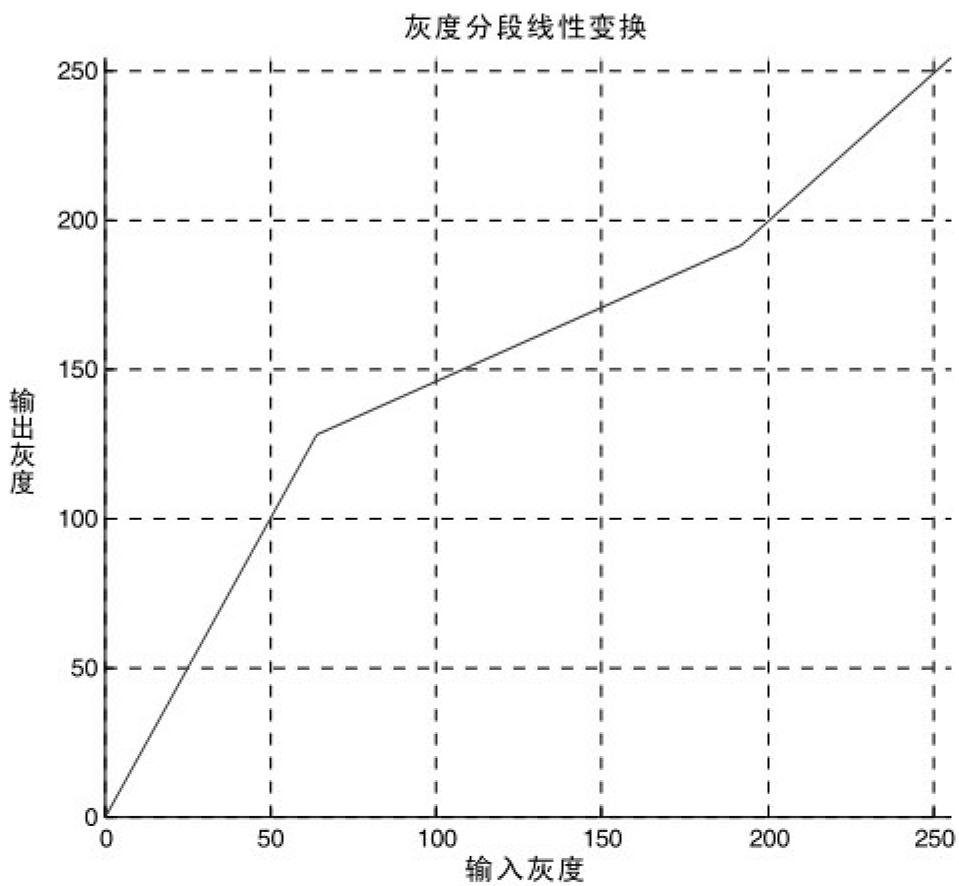
式 3-6 中最重要的参数是$[x_1,x_2]$和$[y_1,y_2]$。根据算法函数描述发现，其中x_1和x_2是给出需要转换的灰度范围，y_1和y_2参数决定线性变换的斜率。

当x_1、x_2、y_1、y_2分别取不同的值的组合时，可得到不同的变换效果。

如果$x_1=y_1$，$x_2=y_2$，则f(x)为1条斜率为1的直线，增强图像将和原图像相同。

● 如果 $x_1=x_2$，$y_1=0$，$y_2=255$，则增强图像只剩下 2 个灰度等级，分段线性变换起到了阈值化的效果，此时的对比度最大，但是细节丢失最多。

● x_1、x_2、y_1、y_2取一般值时的分段线性变换函数图形如图3.14所示。



▲图3.14 灰度的分段线性变换

分段的灰度拉伸可以更加灵活地控制输出灰度直方图的分布，可以有选择地拉伸某段灰度区间以改善输出图像。如果一幅图像灰度集中在较暗的区域而导致图像偏暗，我们可以用灰度拉伸功能来扩展（斜率>1）物体灰度区间以改善图像；同样，如果图像灰度集中在较亮的区域而导致图像偏亮，也可以用灰度拉伸功能来压缩（斜率<1）物体灰度区间以改善图像质量。

灰度拉伸是通过控制输出图像中灰度级的展开程度来达到控制对比度的效果的。一般情况下都限制$x_1<x_2$，$y_1<y_2$，从而保证函数是单调递增的，以避免造成处理过的图像中灰度级发生颠倒。

3.6.2 Matlab实现

我们编写了imgrayscaling函数来实现灰度线性变换，它位于金羽图书论坛（http://bbs.book95.com）的“金羽图书与答疑”板块与本书同名的主题帖子附件中“chapter3/code/”路径下的imgrayscaling.m文件中。

提示

作为本书的第1个由我们自己编写的Matlab图像处理函数，imgrayscaling具有很好的兼容性，它可以处理灰度、彩色、索引等不同类型，以及double、uint8等不同存储方式的图像。出于篇幅考虑，本书后面的所有Matlab函数都只将注意力集中在算法本身，不再考虑对各种类型图像的兼容性问题。如果读者需要处理多种类型的图像，可以参考imgrayscaling函数的编写方法和技巧。

1．输入的处理

为了可以使用可变个数的参数，imgrayscaling函数中使用了细胞数组（参见2.1.6小节），将函数的输入参数整体看做一个细胞数组。为此需要编写一个parse_inputs函数来解析该细胞数组的内容，该函数的返回值为imgrayscaling中所有可能由用户初始化的参数值。

parse_inputs函数的完整实现如下。其中对未知个数的输入使用了参数varargin来表示。

```
function [A, map, x1, x2, y1, y2] = parse_inputs(varargin)
% 这就是用来分析输入参数个数和有效性的函数parse_inputs
% A   输入图像，RGB图（3D），灰度图（2D），或者索引图（X）
% map   索引图调色板（:,3）
% [x1,x2] 参数组 1，曲线中两个转折点的横坐标
% [y1,y2] 参数组 2，曲线中两个转折点的纵坐标
% 首先建立一个空的map变量，以免后面调用isempty(map)时出错
map = [];
% IPTCHECKNARGIN(LOW,HIGH,NUM_INPUTS,FUNC_NAME)
```

检查输入参数的个数是否

% 符合要求，即NUM_INPUTS中包含的输入变量个数是否在LOW和HIGH所指定的范围

% 内。如果不在范围内，则此函数给出一个格式化的错误信息。

iptchecknargin(3,4,nargin,mfilename);

%
IPTCHECKINPUT(A,CLASSES,ATTRIBUTES,FUNC_NAME,VAR_NAMARG_POS) 检查给定

% 矩阵A中的元素是否属于给定的类型列表。如果存在元素不属于给定的类型，则给出

% 一个格式化的错误信息。

iptcheckinput(varargin{1},...

{'uint8','uint16','int16','double'}, ...

{'real', 'nonsparse'},mfilename,'I, X or RGB',1);

% 根据参数个数的不同，分别确定相应的返回值

switch nargin

case3%　可能是imgrayscaling(I,[x1,x2],[y1,y2]) 或 imgrayscaling(RGB,[x1,x2],

[y1,y2])

A = varargin{1};

x1 = varargin{2}(1);

x2 = varargin{2}(2);

y1 = varargin{3}(1);

y2 = varargin{3}(2);

case 4

A = varargin{1};%　　imgrayscaling(X, map, [x1,x2], [y1,y2])

map = varargin{2};

```
        x1 = varargin{2}(1);
        x2 = varargin{2}(2);
        y1 = varargin{3}(1);
        y2 = varargin{3}(2);
        end
        % 检测输入参数的有效性
        % 检查RGB数组
        if (ndims(A)==3) && (size(A,3)～=3)
            msg = sprintf('%s: 真彩色图像应当使用一个M-N-3维度的数组', ...
                  upper(mfilename));
            eid =
sprintf('Images:%s:trueColorRgbImageMustBeMbyNby3',mfilename);
            error(eid,'%s',msg);
        end
        if ～isempty(map)
        % 检查调色板
         if (size(map,2) ～= 3) || ndims(map)>2
            msg1 = sprintf('%s: 输入的调色板应当是一个矩阵', ...
                upper(mfilename));
            msg2 = '并拥有三列';
            eid =
sprintf('Images:%s:inColormapMustBe2Dwith3Cols',mfilename);
            error(eid,'%s %s',msg1,msg2);
         elseif (min(map(:))<0) || (max(map(:))>1)
           msg1 = sprintf('%s: 调色板中各个分量的强度 ',upper(mfilename));
           msg2 = '应当在0和1之间';
            eid = sprintf('Images:%s:colormapValsMustBe0to1',mfilename);
```

```
    error(eid,'%s %s',msg1,msg2);
  end
end
% 将int16类型的矩阵转换成uint16类型
if isa(A,'int16')
  A = int16touint16(A);
end
```

2．输出的处理

可以直接通过nargout这个参数判断用于接收结果的参数个数。参数nargout是由Matlab自动赋值的，如果调用imgrayscaling时没有使用变量接收返回值，则函数直接将结果通过imshow命令显示出来。

```
% 输出
if nargout==0 % 显示结果
  imshow(out);
  return;
end
```

3．函数实现

imgrayscaling函数的完整实现如下。

```
function out = imgrayscaling(varargin)
% IMGRAYSCALING    执行灰度拉伸功能
%   语法：
%       out = imgrayscaling(I, [x1,x2], [y1,y2]);
%       out = imgrayscaling(X, map, [x1,x2], [y1,y2]);
%       out = imgrayscaling(RGB, [x1,x2], [y1,y2]);
%   这个函数提供灰度拉伸功能，输入图像应当是灰度图像，但如果提供的不是灰度
%   图像的话，函数会自动将图像转化为灰度形式。x1，x2，y1，
```

y2应当使用双精度

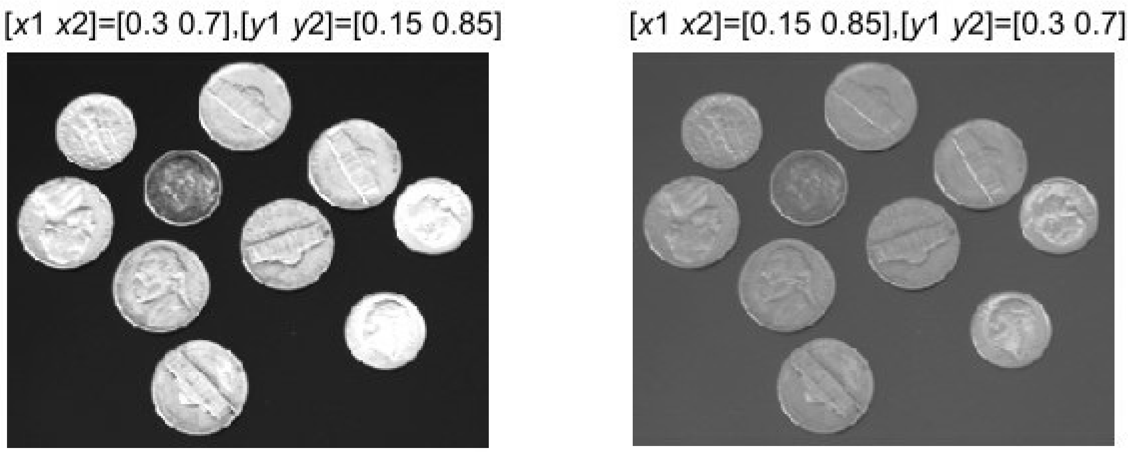
```
%   类型存储，图像矩阵可以使用任何Matlab支持的类型存储。
[A, map, x1 , x2, y1, y2] = parse_inputs(varargin{:});
% 计算输入图像A中数据类型对应的取值范围
range = getrangefromclass(A);
range = range(2);
% 如果输入图像不是灰度图，则需要执行转换
if ndims(A)==3,% A矩阵为3维，RGB图像
   A = rgb2gray(A);
elseif ～isempty(map),% MAP变量为非空，索引图像
   A = ind2gray(A,map);
end % 对灰度图像则不需要转换
% 读取原始图像的大小并初始化输出图像
[M,N] = size(A);
I = im2double(A);   将输入图像转换为双精度类型
out = zeros(M,N);
% 主体部分，双级嵌套循环和选择结构
  for i=1:M
    for j=1:N
       if I(i,j)<x1
          out(i,j) = y1 * I(i,j) / x1;
       elseif I(i,j)>x2
          out(i,j) = (I(i,j)-x2)*(range-y2)/(range-x2) + y2;
       else
          out(i,j) = (I(i,j)-x1)*(y2-y1)/(x2-x1) + y1;
        end
      end
```

```
    end
  % 将输出图像的格式转化为与输入图像相同
  if isa(A, 'uint8') % uint8
    out = im2uint8(out);
  elseif isa(A, 'uint16')
    out = im2uint16(out);
  % 其他情况，输出双精度类型的图像
  end
  % 输出:
  if nargout==0 % 如果没有提供参数接受返回值
    imshow(out);
    return;
  end
```

下面给出使用 imgrayscaling函数实现灰度线性变换的调用示例，实现中分别使用了 2组不同的变换参数。

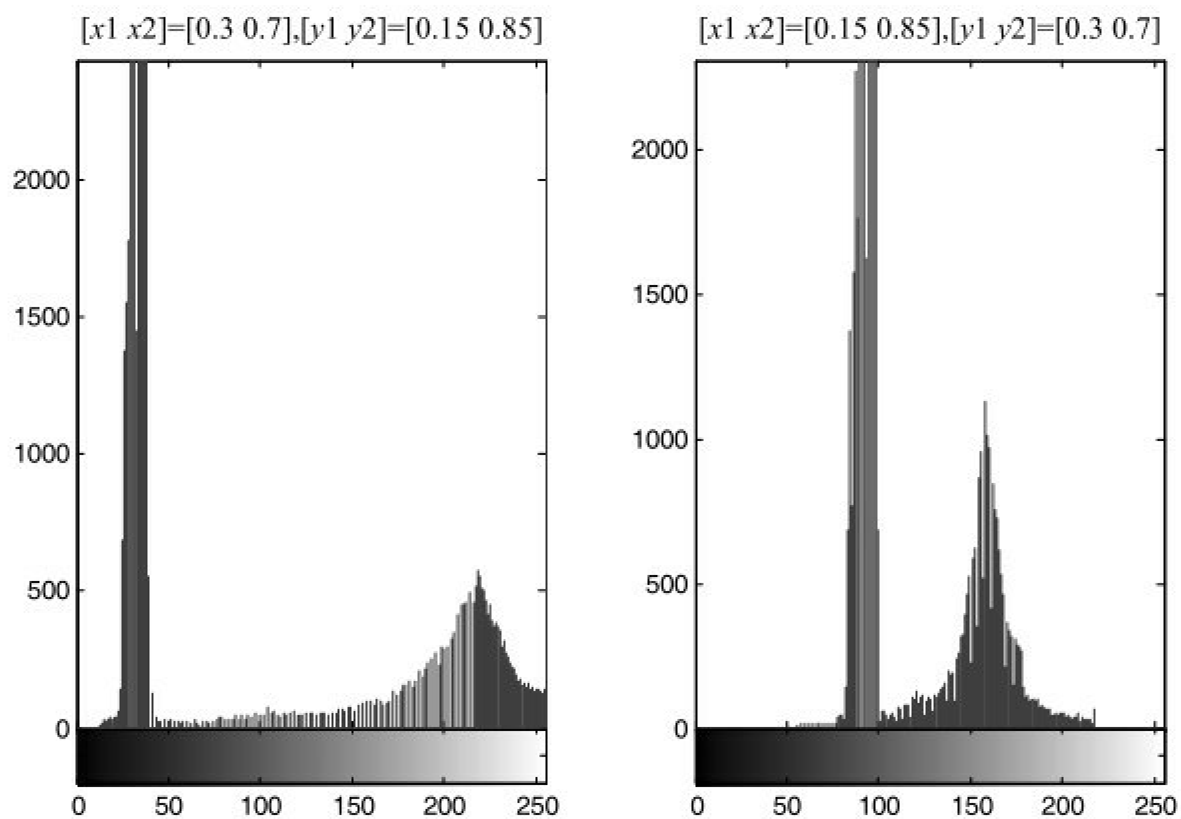
```
>> I = imread('coins.png'); %读入原图像
>> J1 = imgrayscaling(I, [0.3 0.7], [0.15 0.85]);
>> figure, imshow(J1, []);%得到图3.15左图
>> J2 = imgrayscaling(I, [0.15 0.85], [0.3 0.7]);
>> figure, imshow(J2, []); %得到图3.15右图
```

上述程序的运行结果如图3.15和图3.16所示。



▲图3.15 对图 3.7（a）的分段灰度变换的效果

从图3.15和图3.16可以看到第一组参数让图像灰度直方图上的非零区域扩展，而第二组参数让图像的灰度直方图非零区域压缩，这给目标图像带来了截然不同的效果。第一幅图像中的细节更加清晰，而第二幅图像更加柔和。



▲图3.16 分段灰度变换对直方图的影响

3.7 直方图均衡化

本节将介绍一种实用性极高的直方图修正技术——直方图均衡化。

3.7.1 理论基础

直方图均衡化又称为灰度均衡化，是指通过某种灰度映射使输入图像转换为在每一灰度级上都有近似相同的像素点数的输出图像（输出的直方图是均匀的）。在经过均衡化处理后的图像中，像素将占有尽可能多的灰度级并且分布均匀。因此，这样的图像将具有较高的对比度和较大的动态范围。

为了便于分析，我们首先考虑灰度范围为0～1且连续的情况。此时图像的归一化直方图即为概率密度函数（PDF）。

$$p(x), 0 \leqslant x \leqslant 1 \tag{3-7}$$

由概率密度函数的性质，有

$$\int_{x=0}^{1} p(x) = 1 \tag{3-8}$$

设转换前图像的概率密度函数为 $p_r(r)$，转换后图像的概率密度函数为 $p_s(s)$，转换函数（灰度映射关系）为s=f(r)。由概率论知识可得

$$P_s(s) = P_r(r) \cdot \frac{\mathrm{d}r}{\mathrm{d}s} \tag{3-9}$$

这样，如果想使转换后图像的概率密度函数 $p_s(S)=1$，$0 \le S \le 1$（即直

方图为均匀的），则必须满足

$$P_r(r)=\frac{\mathrm{d}r}{\mathrm{d}s} \tag{3-10}$$

等式两边对r积分，可得

$$s=f(r)=\int_0^r p_r(\mu)\mathrm{d}\mu \tag{3-11}$$

式3-11称为图像的累积分布函数（CDF）。

式 3-11是在灰度取值在[0, 1]范围内的情况下推导出来的，对于[0, 255]的情况，只要乘以最大灰度值D_{max}（对于灰度图就是255）即可。此时灰度均衡的转换公式为

$$D_B=f(D_A)=D_{\max}\int_0^{D_A} p_{D_A}(\mu)\mathrm{d}\mu \tag{3-12}$$

式中：D_B为转换后的灰度值；D_A为转换前的灰度值。

而对于离散灰度级，相应的转换公式应为

$$D_B=f(D_A)=\frac{D_{\max}}{A_0}\sum_{i=0}^{D_A}H_i \tag{3-13}$$

式中：H_i为第i级灰度的像素个数；A_0为图像的面积，即像素总数。

式 3-15 中的变换函数f是一个单调增加的函数，这保证了在输出图像中不会出现灰度反转的情况（变换后相对灰度不变），从而能够防止在变换中改变图像的实质，以至于影响对图像的识别和判读。

这里还需要说明一点，对于式 3-13 的离散变换，通常无法再像连

续变换时那样可以得到严格的均匀概率密度函数（$p_s(S)=1$，$0\leq S\leq 1$）。但无论如何，式 3-13 的应用有展开输入图像直方图的一般趋势，可使得均衡化过的图像灰度级具有更大的范围，从而得到近似均匀的直方图。

3.7.2 Matlab实现

Matlab图像处理工具箱提供了用于直方图均衡化的函数histeq，调用语法为

[J, T] = histeq(I)

参数说明

- I是原始图像。

返回值

- J是经过直方图均衡化的输出图像。
- T是变换矩阵。

图像易受光照、视角、方位、噪声等的影响。在这些因素的作用下，同一类图像的不同变形体之间的差距有时大于该类图像与另一类图像之间的差距，这就给图像识别/分类带来了困扰。图像归一化就是将图像转换成唯一的标准形式以抵抗各种变换，从而可消除同类图像不同变形体之间的外观差异。

当图像归一化用于消除灰度因素（光照等）造成的图像外观变化时，称为（图像）灰度归一化。例3.3为我们展示了如何利用直方图均衡化来实现图像的灰度归一化。

[例3.3]利用直方图均衡化技术实现图像的灰度归一化。

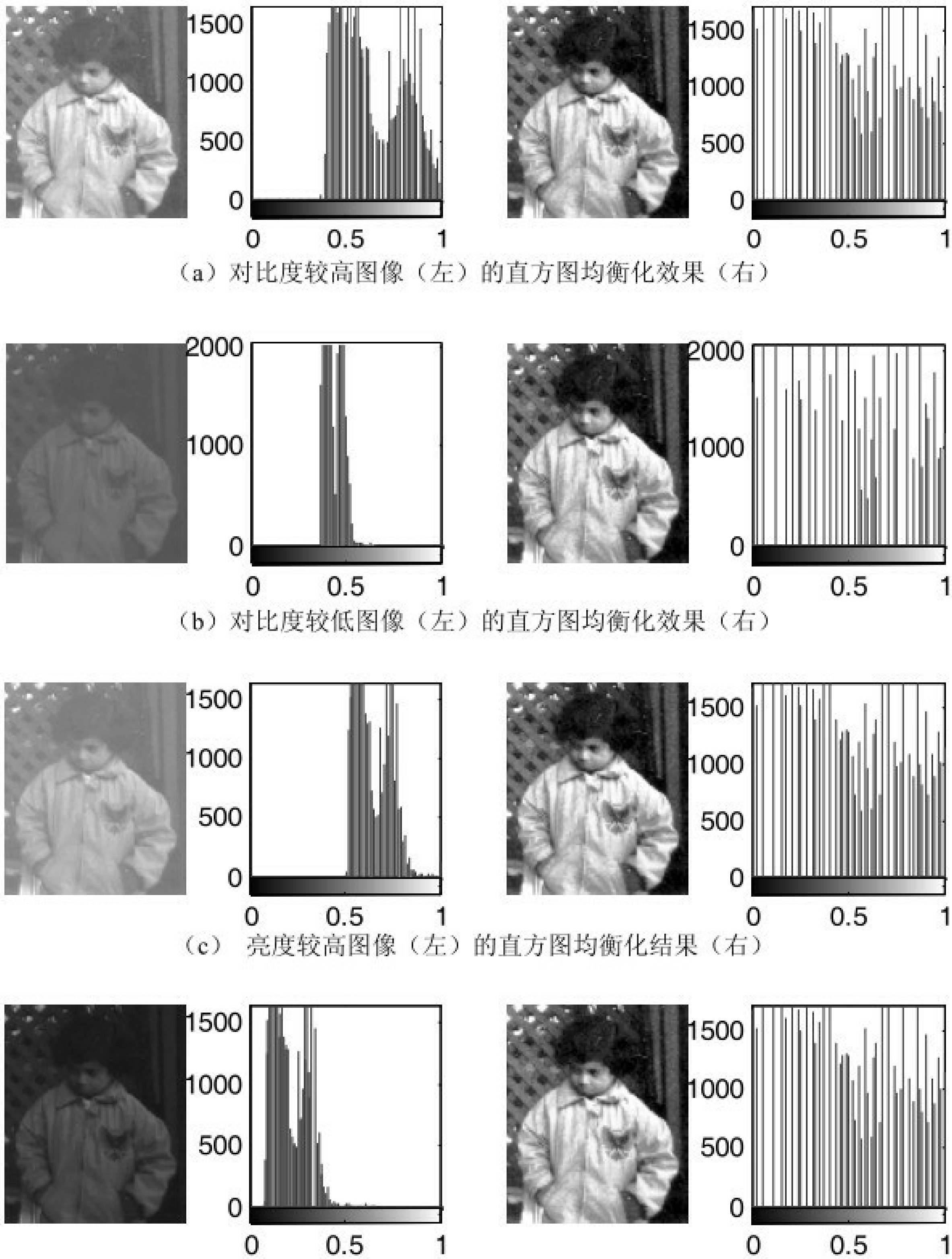
下面的程序在读入了图像pout.tif后，分别对其进行了增加对比度、减小对比度、线性增加亮度和线性减小亮度的处理，得到了原图像的4个灰度变化版本；接着又分别对这4幅图像进行了直方图均衡化处理并

显示了他们在处理前、后的直方图。

```
I = imread('pout.tif'); %读入原图像
I = im2double(I);
% 对于对比度变大的图像
I1 = 2 * I - 55/255;
subplot(4,4,1);
imshow(I1);
subplot(4,4,2);
imhist(I1);
subplot(4,4,3);
imshow(histeq(I1));
subplot(4,4,4);
imhist(histeq(I1));
% 对于对比度变小的图像
I2 = 0.5 * I + 55/255;
subplot(4,4,5);
imshow(I2);
subplot(4,4,6);
imhist(I2);
subplot(4,4,7);
imshow(histeq(I2));
subplot(4,4,8);
imhist(histeq(I2));
% 对于线性增加亮度的图像
I3 = I + 55/255;
subplot(4,4,9);
imshow(I3);
```

```
subplot(4,4,10);
imhist(I3);
subplot(4,4,11);
imshow(histeq(I3));
subplot(4,4,12);
imhist(histeq(I3));
% 对于线性减小亮度的图像
I4 = I - 55/255;
subplot(4,4,13);
imshow(I4);
subplot(4,4,14);
imhist(I4);
subplot(4,4,15);
imshow(histeq(I4));
subplot(4,4,16);
imhist(histeq(I4));
```

上述程序的运行的结果如图3.17所示。



（a）对比度较高图像（左）的直方图均衡化效果（右）

（b）对比度较低图像（左）的直方图均衡化效果（右）

（c） 亮度较高图像（左）的直方图均衡化结果（右）

（d）亮度较低图像（左）的直方图均衡化结果（右）

▲图3.17 不同亮度、对比度图像的直方图均衡化效果

从图 3.17 中我们发现，将直方图均衡化算法应用于左侧的亮度、对比度不同的各个图像后，得到了右侧直方图大致相同的图像，这体现了直方图均衡化作为强大自适应性的增强工具的作用。当原始图像的直方图不同而图像结构性内容相同时，直方图均衡化所得到的结果在视觉上几乎是完全一致的。这对于在进行图像分析和比较之前将图像转化为统一的形式是十分有益的。

从灰度直方图的意义上说，如果一幅图像的直方图非零范围占有所有可能的灰度级并且在这些灰度级上均匀分布，那么这幅图像的对比度较高，而且灰度色调较为丰富，从而易于进行判读。直方图均衡化算法恰恰能满足这一要求。

本书后面将要介绍的很多图像处理方法都是以归一化为目的的，比如第4章几何变换就是研究一种几何失真的归一化处理。

3.8 直方图规定化

直方图均衡化算法可以自动确定灰度变换函数，从而获得具有均匀直方图的输出图像。它主要用于增强动态范围偏小的图像对比度，丰富图像的灰度级。这种方法的优点是操作简单，且结果可以预知，当图像需要自动增强时是一种不错的选择。

但有时希望可以对变换过程加以控制，如能够人为地修正直方图的形状，或者说是获得具有指定直方图的输出图像。这样就可以有选择地增强某个灰度范围内的对比度或使图像灰度值满足某种特定的分布。这种用于产生具有特定直方图的图像的方法叫做直方图规定化，或直方图匹配。

3.8.1 理论基础

直方图规定化是在运用均衡化原理的基础上，通过建立原始图像和期望图像（待匹配直方图的图像）之间的关系，使原始图像的直方图匹配特定的形状，从而弥补了直方图均衡不具备交互作用的特性。

其匹配原理是先对原始的图像均衡化。

$$s = f(r) = \int_0^r p_r(\mu)\mathrm{d}\mu \tag{3-14}$$

同时对待匹配直方图的图像进行均衡化处理。

$$v = g(z) = \int_0^z p_z(\lambda)\mathrm{d}\lambda \; V \tag{3-15}$$

由于都是均衡化，故可令 s = v，则

$$v = g^{-1}(s) = g^{-1}(f(r)) \tag{3-16}$$

于是我们可以按照如下的步骤由输入图像得到一个具有规定概率密度函数的图像。

（1）根据式3-14得到变换关系f(r)。

（2）根据式3-15得到变换关系g(z)。

（3）求得反变换函数g^{-1} 。

（4）对输入图像所有像素应用式3-16中的变换，从而得到输出图像。

当然，在实际计算中我们利用的是上述公式的离散形式，这样就不必去关心函数 f(r)、g(z)以及反变换函数g^{-1} 具体的解析形式，而可以直接将它们作为映射表处理了。其中，f(r)为输入图像均衡化的离散灰度级映射关系；g(z)为标准图像均衡化的的离散灰度级映射关系；而g^{-1} 则是标准图像均衡化的逆映射关系，它给出了从经过均衡化处理的标准化图像到原标准图像的离散灰度映射，相当于均衡化处理的逆过程。

3.8.2 Matlab实现

histeq 函数不仅可以用于直方图均衡化，也可以用于直方图规定化，此时需要提供可选参数hgram。调用语法如下。

[J, T] = histeq(I, hgram)

函数会将原始图像I处理成一幅以用户指定向量hgram作为直方图的图像。

参数hgram的分量数目即为直方图的收集箱数目。对于double型图像，hgram的元素取值范围是[0, 1]；对于 uint8型图像为[0, 255]；对于uint16型图像则为[0, 65535]。

其他参数的意义与在直方图均衡化中的相同。

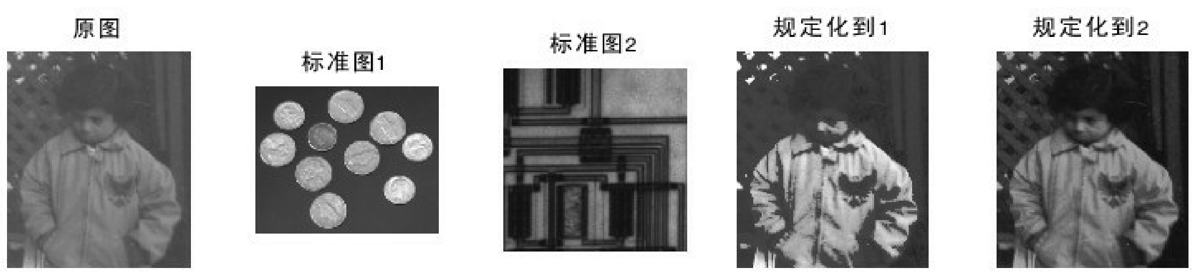
[例3.4]直方图匹配

下面的程序实现了从图像I分别到图像I1和I2的直方图匹配。

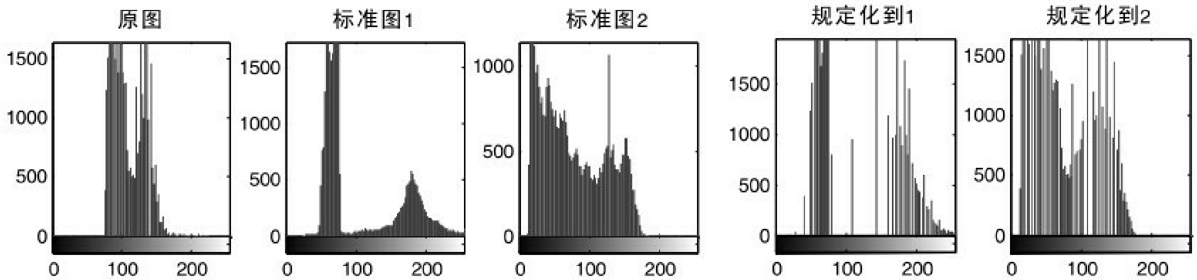
```
I = imread('pout.tif');    %读入原图像
I1 = imread('coins.png');    %读入要匹配直方图的图像
I2 = imread('circuit.tif');    %读入要匹配直方图的图像
% 计算直方图
[hgram1, x] = imhist(I1);
[hgram2, x] = imhist(I2);
% 执行直方图均衡化
J1=histeq(I,hgram1);
J2=histeq(I,hgram2);
% 绘图
subplot(2,3,1);
imshow(I);title('原图');
subplot(2,3,2);
imshow(I1); title('标准图1');
```

```
subplot(2,3,3);
imshow(I2); title('标准图2');
subplot(2,3,5);
imshow(J1); title('规定化到1')
subplot(2,3,6);
imshow(J2);title('规定化到2');
% 绘直方图
figure;
subplot(2,3,1);
imhist(I);title('原图');
subplot(2,3,2);
imhist(I1); title('标准图1');
subplot(2,3,3);
imhist(I2); title('标准图2');
subplot(2,3,5);
imhist(J1); title('规定化到1')
subplot(2,3,6);
imhist(J2);title('规定化到2');
```

上述程序的运行结果如图3.18和图3.19所示。可以看到，经过规定化处理，原图像的直方图与目标图像的直方图变得较为相似。



▲图3.18 直方图规定化结果



▲图3.19 直方图规定化后的灰度直方图

直方图规定化本质上是一种拟合过程，因此变换得到的直方图与标准目标图像的直方图并不会完全一致。然而即使只是相似的拟合，仍然使规定化的图像在亮度与对比度上具有类似标准图像的特性，这正是直方图规定化的目的所在。

第4章 图像的几何变换

包含相同内容的两幅图像可能由于成像角度、透视关系乃至镜头自身原因所造成的几何失真而呈现截然不同的外观，这就给观测者或是图像识别程序带来了困扰。通过适当的几何变换可以最大程度地消除这些几何失真所产生的负面影响，有利于我们在后续的处理和识别工作中将注意力集中于图像的内容本身，更确切地说是图像中的对象，而不是该对象的角度和位置等。因此，几何变换常常作为其他图像处理应用的预处理步骤，是图像归一化的核心工作之一。

本章的知识和技术热点

- 图像的各种几何变换，如平移、镜像和旋转等
- 插值算法
- 图像配准

本章的典型案例分析

- 人脸图像的配准

4.1 解决几何变换的一般思路

图像几何变换又称为图像空间变换，它将一幅图像中的坐标位置映射到另一幅图像中的新坐标位置。我们学习几何变换的关键就是要确定这种空间映射关系，以及映射过程中的变换参数。

几何变换不改变图像的像素值，只是在图像平面上进行像素的重新安排。一个几何变换需要两部分运算：首先是空间变换所需的运算，如平移、旋转和镜像等，需要用它来表示输出图像与输入图像之间的（像素）映射关系；此外，还需要使用灰度插值算法，因为按照这种变换关系进行计算，输出图像的像素可能被映射到输入图像的非整数坐标上。

设原图像 $f(x_0,y_0)$经过几何变换产生的目标图像为 $g(x_1,y_1)$，则该空间变换（映射）关系可表示为

$$x_1 = s(x_0, y_0) \tag{4-1}$$

$$y_1 = t(x_0, y_0) \tag{4-2}$$

其中，$s(x_0,y_0)$和 $t(x_0,y_0)$为由 $f(x_0,y_0)$到$g(x_1,y_1)$的坐标变换函数。例如，当$x_1= s(x_0,y_0)= 2x_0$，$y_1= t(x_0,y_0)=2y_0$时，变换后的图像 $g(x_1,y_1)$只是简单地在 x和 y两个空间方向上将 $f(x_0,y_0)$的尺寸放大一倍。因此，我们看到只要掌握了有关变换函数$s(x_0,y_0)$和$t(x_0,y_0)$的情况，就可以遵循下面的步骤实现几何变换。

算法**4.1**

根据空间变换的映射关系，确定变换后目标图像的大小（行、列范围）；//有些变换可能改变图像大小

计算逆变换$s^{-1}(j_1,i_1)$和$t^{-1}(j_1,i_1)$；

逐行扫描目标图像$g(x_1,y_1)$，对于$g(x_1,y_1)$中的每一点(j_0,i_0)：

{

根据空间变换的映射关系，计算得：

$j_0'= s^{-1}(j_1,i_1)$; //直接通过映射关系计算得到的横坐标，可能不是整数

$i_0'= t^{-1}(j_1,i_1)$; //直接通过映射关系计算得到的纵坐标，可能不是整数

根据选用的插值方法：

$(j_0,i_0) = interp(j_0', i_0')$; //对于非整数坐标（$j_0'$, i_0'）需要插值

If (j_0,i_0) 在图像f之内

拷贝对应像素：$g(j_1,i_1) = f(j_0,i_0)$；

Else

$g(j_1,i_1) = 255;$

}

对于几何失真图像的复原（校正）过程正好是上述变换的逆过程。

$$x_0 = s^{-1}(x_1, y_1) \quad （4-3）$$

$$y_0 = t^{-1}(x_1, y_1) \quad （4-4）$$

式 4-3和式4-4表示相应的由 $g(x_1,y_1)$到 $f(x_0,y_0)$的逆变换。此时，经过某种几何变换而失真的图像 $g(x_1,y_1)$是我们要复原的对象，原始图像 $f(x_0,y_0)$是我们复原的目标。

对服务于识别的图像处理而言，作为图像几何归一化的逆变换过程的应用常常更为广泛。当然，在变换中究竟以谁作为原始图像 $f(x_0,y_0)$，以谁作为变换图像$g(x_1,y_1)$并不是绝对的，这完全取决于我们在分析特定问题过程中的立场。比如说对于图4.1中的两幅图像，一般的做法是以图4.1（a）为原始图像，图4.1（b）为变换图像。这是因为在图4.1（a）中我们关心的对象（数字和字母）处于一个便于观察的角度（正的）。但我们也完全可以将图 4.1（b）视为 $f(x_0,y_0)$，而图 4.1（a）为$g(x_1,y_1)$。此时，相应的映射关系s和t也会发生变化。

注意

当图像归一化（参见3.7节）用于消除几何因素（视角、方位等）造成的图像外观变化时，称为（图像）几何归一化，它能够排除对象间几何关系的差别，找出图像中的那些几何不变量，从而得知这些对象原本就是一样的或属于相同的类别。



（a）



（b）

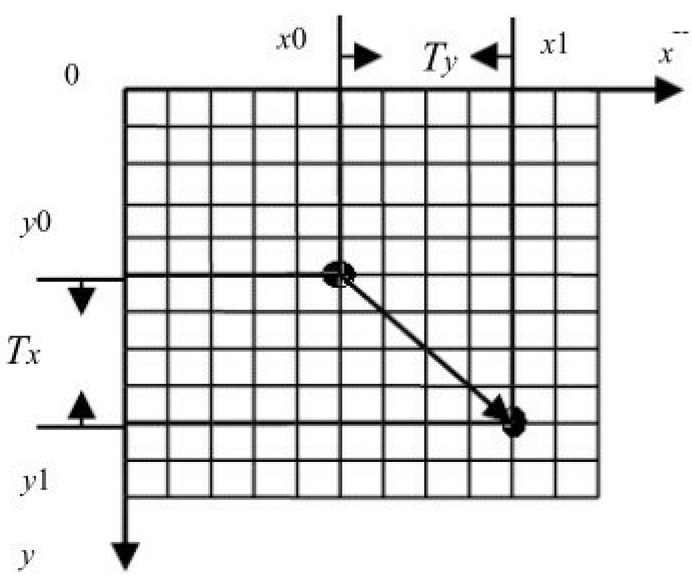
▲图4.1 旋转前后的两幅图像

4.2 图像平移

图像平移就是将图像中所有的点按照指定的平移量水平或者垂直移动。

4.2.1 图像平移的变换公式

设（x_0, y_0）为原图像上的一点，图像水平平移量为t_x，垂直平移量为t_y，如图 4.2所示。



▲图4.2 平移变换坐标图

则平移之后的点坐标（$x_{1,}y_1$）变为

$$\begin{cases} x_1 = x_0 + T_x \\ y_1 = y_0 + T_y \end{cases}$$

用矩阵表示为

$$[\,x_1 \quad y_1 \quad 1]=[\,x_0 \quad y_0 \quad 1]\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ T_x & T_y & 1 \end{pmatrix}$$

对变换矩阵求逆，可以得到逆变换

$$[\,x_0 \quad y_0 \quad 1]=[\,x_1 \quad y_1 \quad 1]\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -T_x & -T_y & 1 \end{pmatrix}$$

即 $$\begin{cases} x_0 = x_1 - T_x \\ y_0 = y_1 - T_y \end{cases}$$

这样，平移后的目标图像中的每一点都可以在原图像中找到对应的点。例如，对于新图中的（i, j）像素，代入上面的方程组，可以求出对应原图中的像素（$i-t_x$，$j-t_y$）。而此时如果t_x大于i或t_y大于j，则点（$i-t_x$，$j-t_y$）超出了原图的范围，可以直接将它的像素值统一设置为0或者255。

对于原图中被移出图像显示区域的点通常也有两种处理方法，可以直接丢弃，也可以通过适当增加目标图像的尺寸（将新生成的图像宽度

增加t_x，高度增加t_y）的方法使得新图像中能够包含这些点。在稍后给出的程序实现中，我们采用了第一种处理方法。

4.2.2 图像平移的Matlab实现

imtransform 函数用于完成一般的二维空间变换，本章后续介绍的很多几何变换都可以通过该函数来实现，原型如下。

B = imtransform(A,TFORM,method);

- 参数A为要进行几何变换的图像。
- 空间变换结构TFORM指定了具体的变换类型。
- 可选参数method允许为imtransform函数选择插值方法，其合法值如表4.1所示。

表4.1 合法值

合 法 值	含 义
'bicubic'	双三次插值
'bilinear'	双线性插值
'nearest'	最近邻插值

这些插值方式的具体含义请参见4.7节。默认时为双线性插值—'bilinear'。

函数输出B为经imtransform变换后的目标图像。

可以通过两种方法来创建TFORM结构，即使用maketform函数和cp2tform函数。cp2tform是一个数据拟合函数，它需要原图像与目标图像之间的对应点对作为输入，用于确定基于控制点对的几何变换关系，我们将在 4.8 节图像配准中对它进行介绍，这里仅给出使用 maketform 函数获得TFORM结构的方法。

T=maketform(transformtype, Matrix);

● 参数transformtype指定了变换的类型，如常见的'affine'为二维或多维仿射变换，包括平移、旋转、比例、拉伸和错切等。

● Matrix为相应的仿射变换矩阵，例如对于平移变换，该矩阵为

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ T_x & T_y & 1 \end{pmatrix}$$

。

使用imtransform函数实现图像平移的方法如例4.1所示。

[例4.1]图像平移

本例将图像平移的实现封装在 imMove()函数中，读者可以在金羽图书论坛（ http://bbs. book95.com）的“金羽图书与答疑”板块与本书同名的主题帖子附件的“chapter3/code/imMove.m”中找到它。

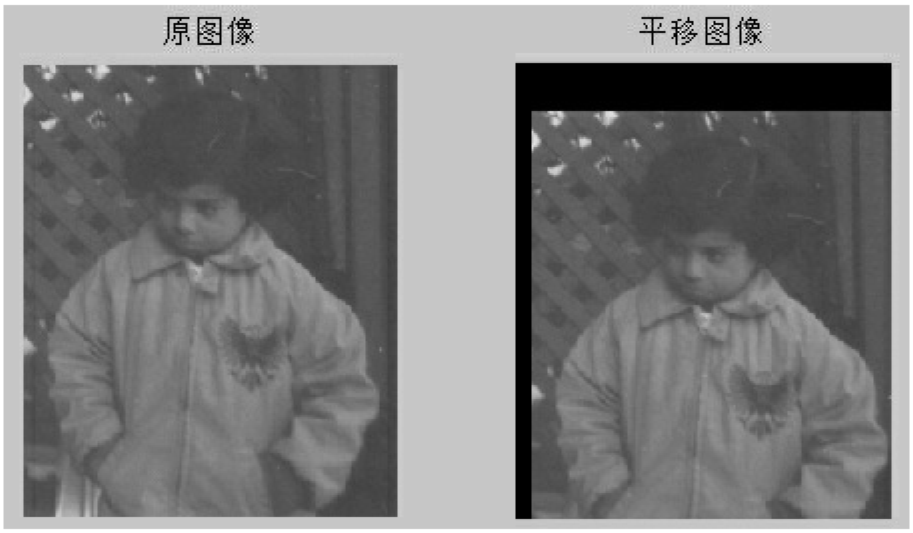
```
function I_out = imMove(I, tx, ty)
% 平移变换
% 输入：I - 输入图像
%       tx- 水平方向平移量
%       ty- 竖直方向平移量
% 输出：I_out - 输出图像
tform = maketform('affine',[1 0 0;0 1 0; tx ty 1]);
%定义平移变换矩阵
I_out = imtransform(I,tform,'XData',[1 size(I,2)],'YData',[1 size(I,1)]);
%图像平移
subplot(1,2,1),imshow(I);
title('原图像');
subplot(1,2,2),imshow(I_out);
title('平移图像');
```

注意

由于平移变换前后的 2 幅图像像素本身及其关系并无变化，导致与输入图像相同尺寸的输出图像I_out在显示时与输入图像I没有任何差别。在imtransform函数中加入了XData和YData选项可以解决之一问题。本章后续的一些几何变换则不需要使用XData与YData选项。

下面的命令调用imMove()函数后的平移效果如图4.3所示，注意到对于映射在原图像之外的点算法直接采用黑色（0）填充，并丢弃了变换后目标图像中被移出图像显示区域的像素。

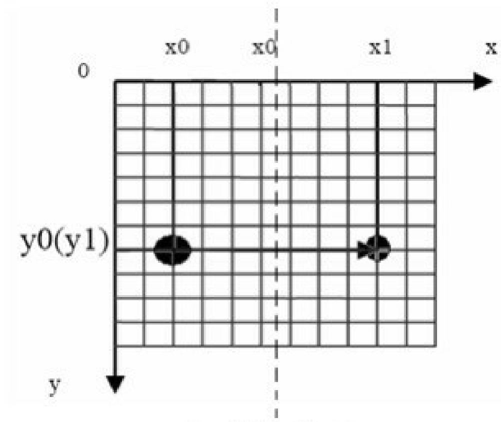
I_out = imMove(I, 10, 30); %水平方向平移10个像素，竖直方向平移30个像素

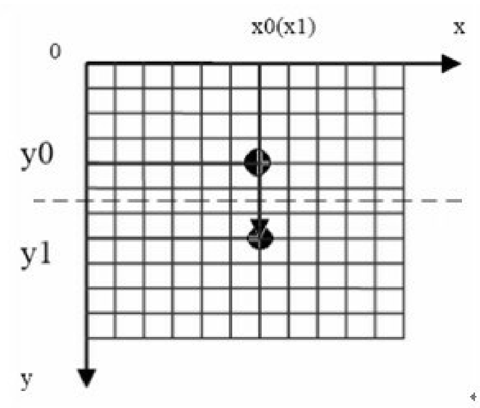


▲图4.3 平移变换效果

4.3 图像镜像

镜像变换又分为水平镜像和竖直镜像。水平镜像即将图像左半部分和右半部分以图像竖直中轴线为中心轴进行对换；而竖直镜像则是将图像上半部分和下半部分以图像水平中轴线为中心轴进行对换，如图4.4所示。





▲图4.4 镜像变换，其中心轴在图中以虚线标出

4.3.1 图像镜像的变换公式

● 水平镜像的变换关系为

$$[x_1 \quad y_1 \quad 1]=[x_0 \quad y_0 \quad 1]\begin{pmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ \text{Width} & 0 & 1 \end{pmatrix}=[\text{Width}-x_0 \quad y_0 \quad 1]$$

对矩阵求逆得到

$$[x_0 \quad y_0 \quad 1]=[x_1 \quad y_1 \quad 1]\begin{pmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ \text{Width} & 0 & 1 \end{pmatrix}=[\text{Width-}x_1 \quad y_1 \quad 1]$$

● 竖直镜像变换关系可形式化地描述如下。

$$[x_1 \quad y_1 \quad 1]=[x_0 \quad y_0 \quad 1]\begin{pmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & \text{Height} & 1 \end{pmatrix}=[x_0 \quad \text{Height-}y_0 \quad 1]$$

逆运算为

$$[x_0 \quad y_0 \quad 1]=[x_1 \quad y_1 \quad 1]\begin{pmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & \text{Height} & 1 \end{pmatrix}=[x_1 \quad \text{Height-}\, y_1 \quad 1]$$

4.3.2 图像镜像的Matlab实现

一个镜像变换的示例程序如例4.2所示。

[例4.2]图像镜像

```
% 镜像变换
A=imread('pout.tif');
[height,width,dim]=size(A);
tform = maketform('affine',[-1 0 0;0 1 0; width 0 1]);
%定义水平镜像变换矩阵
B = imtransform(A,tform,'nearest');
tform2 = maketform('affine',[1 0 0;0 -1 0; 0 height 1]);
%定义竖直镜像变换矩阵
C = imtransform(A,tform2,'nearest');
subplot(1,3,1),imshow(A);
title('原图像');
subplot(1,3,2),imshow(B);
title('水平镜像');
subplot(1,3,3),imshow(C);
title('竖直镜像');
```

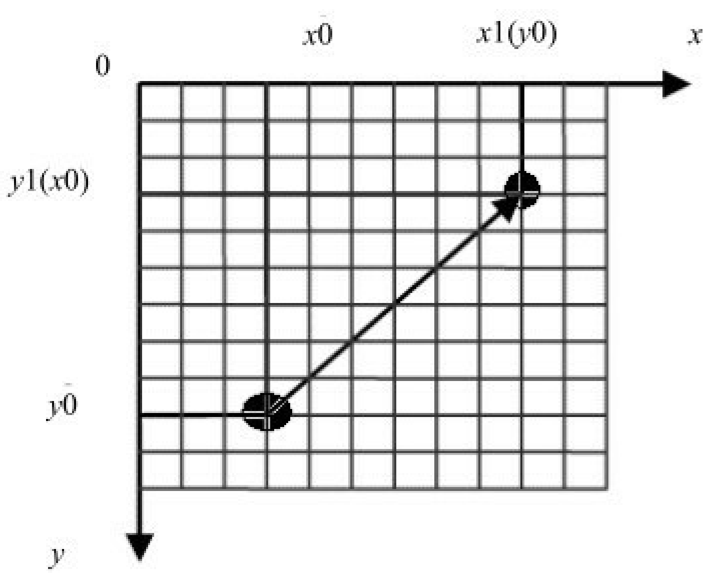
运行结果如图4.5所示。



▲图4.5 镜像变换效果图

4.4 图像转置

图像转置是将图像像素的x坐标和y坐标互换，如图4.6所示。转置后图像的大小会随之改变，高度和宽度将互换。



▲图4.6 转置变换坐标图

4.4.1 图像转置的变换公式

转置变换的公式如下。

$$[x_1 \quad y_1 \quad 1]=[x_0 \quad y_0 \; 1]\begin{pmatrix}0&1&0\\1&0&0\\0&0&1\end{pmatrix}=[y_0 \quad x_0 \quad 1]$$

显然，转置矩阵$\begin{pmatrix}0&1&0\\1&0&0\\0&0&1\end{pmatrix}$的逆矩阵仍为其自身。故转置变换的逆变换具有相同的形式。

4.4.2 图像转置的Matlab实现

转置变换的实现程序如例4.3所示。

[例4.3]图像转置

```
% 图像转置
A=imread('pout.tif');
tform = maketform('affine',[0 1 0;1 0 0; 0 0 1]);
%定义转置变换矩阵
B = imtransform(A,tform,'nearest');
subplot(1,2,1),imshow(A);
title('原图像');
subplot(1,2,2),imshow(B);
title('图像转置');
```

转置结果如图4.7所示。

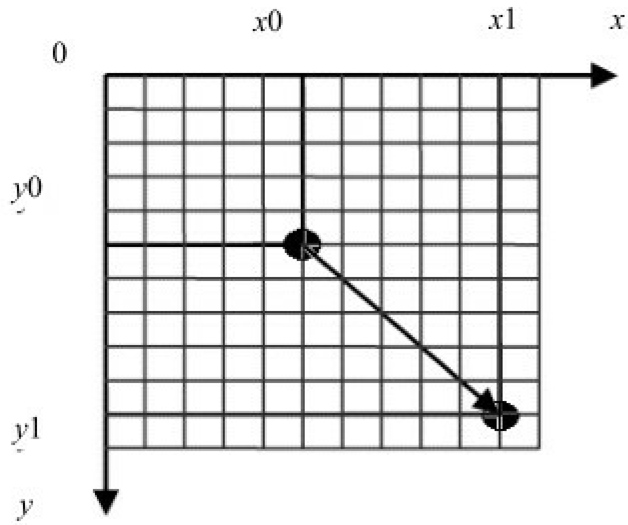
在学习了4.6节图像旋转之后，有兴趣的读者也可尝试通过先水平镜像，再逆时针旋转90°的方式来实现图像转置。



▲图4.7 转置效果图

4.5 图像缩放

图像缩放是指图像按照指定的比率放大或者缩小，如图4.8所示。



▲图4.8 缩放变换效果图

4.5.1 图像缩放的变换公式

假设图像x轴方向的缩放比率S_x，y轴方向的缩放比率S_y，相应的变换表达式为

$$[x_1 \quad y_1 \quad 1]=[x_0 \quad y_0 \quad 1]\begin{pmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{pmatrix}=[x_0 \cdot S_x \quad y_0 \cdot S_y \quad 1]$$

其逆运算如下。

$$[x_0 \quad y_0 \quad 1]=[x_1 \quad y_1 \quad 1]\begin{pmatrix} \frac{1}{S_x} & 0 & 0 \\ 0 & \frac{1}{S_y} & 0 \\ 0 & 0 & 1 \end{pmatrix}=[\frac{x_1}{S_x} \quad \frac{y_1}{S_y} \quad 1]$$

直接根据缩放公式计算得到的目标图像中，某些映射源坐标可能不是整数，从而找不到对应的像素位置。例如，当$S_x=S_y=2$时，图像放大2倍，放大图像中的像素（0, 1）对应于原图中的像素（0, 0.5），这不是整数坐标位置，自然也就无法提取其灰度值。因此我们必须进行某种近似处理，一种简单的策略是直接将它最邻近的整数坐标位置（0,0）或者（0,1）处的像素灰度值赋给它，这就是所谓的最近邻插值。当然还可以通过4.7节将介绍的其他插值算法来近似。

4.5.2 图像缩放的Matlab实现

缩放变换仍然可借助前面几节中使用的imtransform函数来实现。此外，Matlab还提供了专门的图像缩放函数imresize，具体调用形式为

B=imresize(A, Scale, method);

● 参数A为要进行缩放的原始图像。

● Scale为统一的缩放比例。

● 可选参数method用于为imresize指定插值方法，其合法取值同imtransform函数，但默认为最近邻插值。

输出B为缩放后的图像。

例4.4给出了图像等比例缩放的Matlab实现。

[例4.4]图像等比例缩放

```
% 图像缩放
A = imread('pout.tif');
B = imresize(A,1.2,'nearest');
%图像扩大1.2倍
figure,imshow(A);title('原图像');
figure,imshow(B);title('图像缩放');
```

程序运行结果如图4.9所示。



▲图4.9 缩放变换效果图

如果希望在x和y方向上以不同比例进行缩放，可使用如下方式调用imresize函数。

B = imresize(A,[mrows ncols],method);

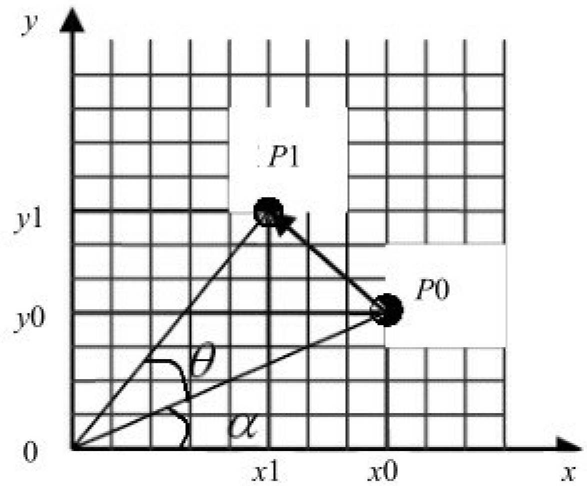
向量参数[mrows ncols]指明了变换后目标图像B的具体行数（高）和列数（宽），其余均与等比例缩放时的调用相同，这里不再赘述。

4.6 图像旋转

旋转一般是指将图像围绕某一指定点旋转一定的角度。旋转通常也会改变图像的大小，和4.2节中图像平移的处理一样，可以把转出显示区域的图像截去，也可以改变输出图像的大小以扩展显示范围。

4.6.1 以原点为中心的图像旋转

如图4.10所示，点P(x_0, y_0)绕原点逆时针旋转角度θ到点P1(x_1, y_1)。



▲图4.10 旋转变换坐标图

令 $L=|\mathrm{OP}|=\sqrt{x^2+y^2}$ ，则有

$$\sin\alpha = y_0/L, \cos\alpha = x_0/L$$

到达P1点后，有

$$\sin(\alpha+\theta)= y_1/\mathrm{L}=\cos\theta\sin\alpha+\sin\theta\cos\alpha$$
$$\cos(\alpha+\theta)= x_1/\mathrm{L}=\cos\theta\cos\alpha-\sin\theta\sin\alpha$$

于是有

$$x_1=\cos\theta\ x_0-\sin\theta\ y_0$$
$$y_1=\cos\theta\ y_0-\sin\theta\ x_0$$

从而得出旋转变换公式为

$$[x_1 \quad y_1 \quad 1]=[x_0 \quad y_0 \quad 1]\begin{pmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

相应的，其逆运算为

$$[x_0 \quad y_0 \quad 1]=[x_1 \quad y_1 \quad 1]\begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

4.6.2 以任意点为中心的图像旋转

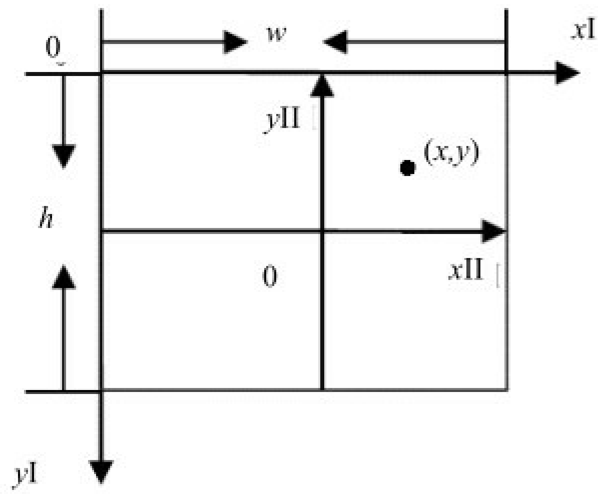
在4.6.1中给出的旋转是以坐标原点为中心进行的，那么如何围绕任意的指定点来旋转呢？将平移和旋转操作相结合即可，即先进行坐标系平移，再以新的坐标原点为中心旋转，之后将新原点平移回原坐标系的原点。这一过程可归纳为以下3个步骤。

（1）将坐标系Ⅰ变成Ⅱ。

（2）将该点顺时针旋转θ角。

（3）将坐标系Ⅱ变回Ⅰ。

下面我们就以围绕图像中心的旋转为例，具体说明上述的变换过程。如图4.11所示，坐标系I以图像左上角点为原点，向右为x轴正方向，向下为y轴正方向；而坐标系II是以图像的中心为原点，向右为x轴正方向，向上为y轴正方向。那么坐标系I与坐标系II之间的转换关系如何呢？



▲图4.11 两种坐标系间的转换关系

设图像的宽为w，高为h，则容易得到

$$\begin{bmatrix} xI \\ yI \\ 1 \end{bmatrix} = \begin{bmatrix} xII \\ yII \\ 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0.5w & 0.5h & 1 \end{bmatrix}$$

相应的逆变换为

$$\begin{bmatrix} xII \\ yII \\ 1 \end{bmatrix} = \begin{bmatrix} xI \\ yI \\ 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ -0.5w & 0.5h & 1 \end{bmatrix}$$

至此，我们已经实现了上述3个步骤中的第1步和第3步，再加上第2步的旋转变换就得到了围绕图像中心点旋转的最终变换矩阵，该矩阵实际上是3个变换步骤中分别用到的3个变换矩阵的级联。

$$\begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} = \begin{bmatrix} x_0 \\ y_0 \\ 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ -0.5\text{Wold} & 0.5\text{Hold} & 1 \end{bmatrix} \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0.5\text{Wnew} & 0.5\text{Hnew} & 1 \end{bmatrix}$$

$$= \begin{bmatrix} x_0 \\ y_0 \\ 1 \end{bmatrix} \begin{bmatrix} \cos(\theta) & \sin(\theta) & 0 \\ -\sin(\theta) & \cos(\theta) & 0 \\ -0.5\ \text{Wold} \cdot \cos(\theta) + 0.5\text{Hold} \cdot \sin(\theta) + 0.5\text{Wnew} & -0.5\text{Wold} \cdot \sin(\theta) - 0.5\text{Hold} \cdot \cos(\theta) + 0.5\text{Hnew} & 1 \end{bmatrix}$$

式中：Wold、Hold、Wnew、Hnew分别表示原图像和新图像的宽和高。

上式的逆变换为

$$\begin{bmatrix} x_0 \\ y_0 \\ 1 \end{bmatrix} = \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ -0.5\text{Wnew} & 0.5\text{Hnew} & 1 \end{bmatrix} \begin{bmatrix} \cos(\theta) & \sin(\theta) & 0 \\ -\sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0.5\text{Wold} & 0.5\text{Hold} & 1 \end{bmatrix}$$

$$= \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ -0.5\text{Wnew} \cdot \cos(\theta) - 0.5\text{Hnew} \cdot \sin(\theta) + 0.5\text{Wold} & 0.5\text{Wnew} \cdot \sin(\theta) - 0.5\text{Hnew} \cdot \cos(\theta) + 0.5\text{Hold} & 1 \end{bmatrix}$$

这样，我们就可以根据上面的逆变换公式，按照算法4.1中的描述来实现围绕图像中心的旋转变换。类似地，可以进一步得出以任意点为中心的旋转变换。

4.6.3 图像旋转的Matlab实现

图像旋转变换的效果受具体插值方法的影响较为明显，本节给出的实现均采用最近邻插值，在4.7节中将给出采用不同的插值算法时图像旋转变换的效果比较。

可通过4.6.2中学习的方法设置适当的变换结构TFORM，从而调用imtransform 来实现以任意点为中心的图像旋转。此外，Matlab还专门提供了围绕图像中心的旋转变换函数imrotate，其调用方式如下。

B=imrotate(A,angle,method, 'crop');

● A是要旋转的图像。

● angle 为旋转角度，单位为度，如为其指定一个正值，则 imrotate 函数按逆时针方向旋转图像。

● 可选参数method为 imrotate函数指定插值方法。

● 'crop'选项会裁剪旋转后增大的图像，使得到的图像和原图大小一致。

下面的例4.5给出了图像旋转的Matlab实现。

[例4.5]图像旋转

```
% 围绕中心点的图像旋转
A=imread('pout.tif');
%最近邻插值法逆旋转30°，并剪切图像
B=imrotate(A,30,'nearest','crop');
subplot(1,2,1),imshow(A);
title('原图像');
subplot(1,2,2),imshow(B);
title('逆时针旋转30°');
```

程序运行结果如图4.12所示。



▲图4.12 旋转变换效果图

4.7 插值算法

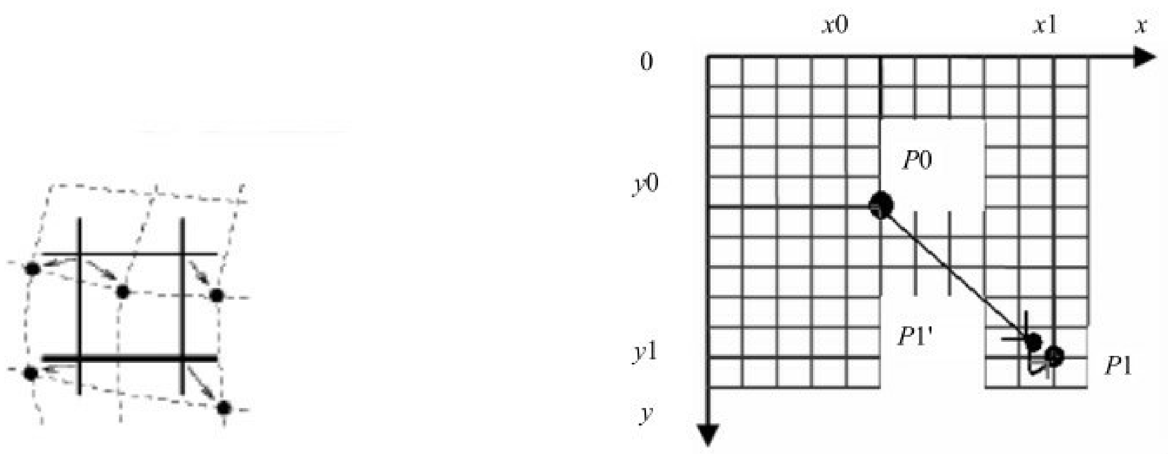
实现几何运算时，有两种方法。第一种称为向前映射法，其原理是将输入图像的灰度一个像素一个像素地转移到输出图像中，即从原图像坐标计算出目标图像坐标：$g(x_1,y_1) = f(a(x_0,y_0), b(x_0,y_0))$。前面的平移、镜像等操作就可以采用这种方法。

另外一种称为向后映射法，它是向前映射变换的逆，即输出像素一个一个地映射回输入图像中。如果一个输出像素映射到的不是输入图像的采样栅格的整数坐标处的像素点，则其灰度值就需要基于整数坐标的灰度值进行推断，这就是插值。由于向后映射法是逐个像素产生输出图像，不会产生计算浪费问题，所以在缩放、旋转等操作中多采用这种方法，本书中采用的也全部为向后映射法。

本节中将介绍3种不同的插值算法，处理效果好的算法一般需要较大的计算量。

4.7.1 最近邻插值

这是一种最简单的插值算法，输出像素的值为输入图像中与其最邻近的采样点的像素值。例如，图4.13中的点P0在几何变换中被映射至点P1'，但由于点P1'处于非整数的坐标位置，无法提取其像素灰度值。所以可以用与P1'最邻近的采样点P1的灰度值近似作为P1'的灰度值。



▲图4.13 最近邻插值示意图

最近邻插值可表示为

$$f(x,y) = g(\text{round}(x), \text{round}(y));$$

最近邻插值计算简单，而且在很多数情况下的输出效果也可以接受。然而，最近邻插值法会在图像中产生人为加工的痕迹，详见例4.6。而我们在之前各种变换中采用的均为双线性插值，它是imtransform函数的默认插值方式。

4.7.2 双线性插值

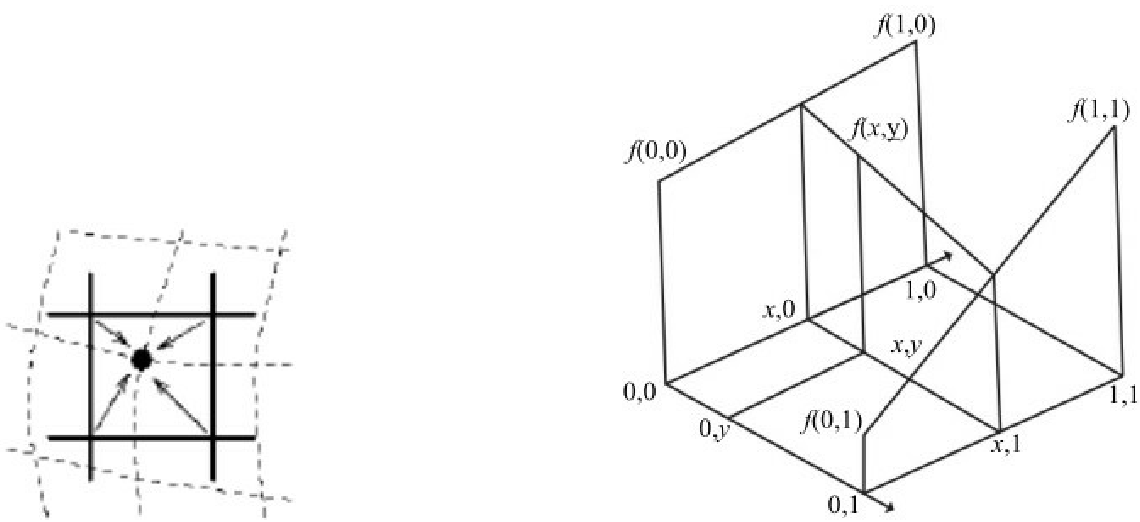
双线性插值又称为一阶插值，是线性插值扩展到二维的一种应用。它可以通过一系列的一阶线性插值得到。

注意

线性（linear），指量与量之间按比例、成直线的关系，在数学上可以理解为一阶导数为常数的函数。线性插值则是指根据两个点的值线性地确定位于这两个点连线上的某一点的值。

输出像素的值为输入图像中距离它最近的2×2邻域内采样点像素灰度值的加权平均。

设已知单位正方形的顶点坐标分别为 f(0,0)、f(1,0)、f(0,1)、f(1,1)，如图4.14 所示，我们要通过双线性插值得到正方形内任意点f(x,y)的值。



▲图4.14 线性插值示意图

首先对上端的两个点进行线性插值，得到

$$f(x,0)=f(0,0)+x[f(1,0)-f(0,0)] \quad (4\text{-}5)$$

再对下端的两个顶点进行线性插值，得到

$$f(x,1)=f(0,1)+x[f(1,1)-f(0,1)] \quad (4\text{-}6)$$

最后，对垂直方向进行线性插值，得到

$$f(x,y)=f(x,y)+y[f(x,1)-f(x,0)] \quad (4\text{-}7)$$

综合式4-5至4-7，整理得出

$$f(x,y)=[f(1,0)-f(0,0)]x+[f(0,1)-f(0,0)]y+[f(1,1)+f(0,0)-f(0,1)-f(1,0)]xy+f(0,0) \quad (4\text{-}8)$$

上面的推导是在单位正方形的前提下进行的，稍加变换就可以推广到一般情况中。

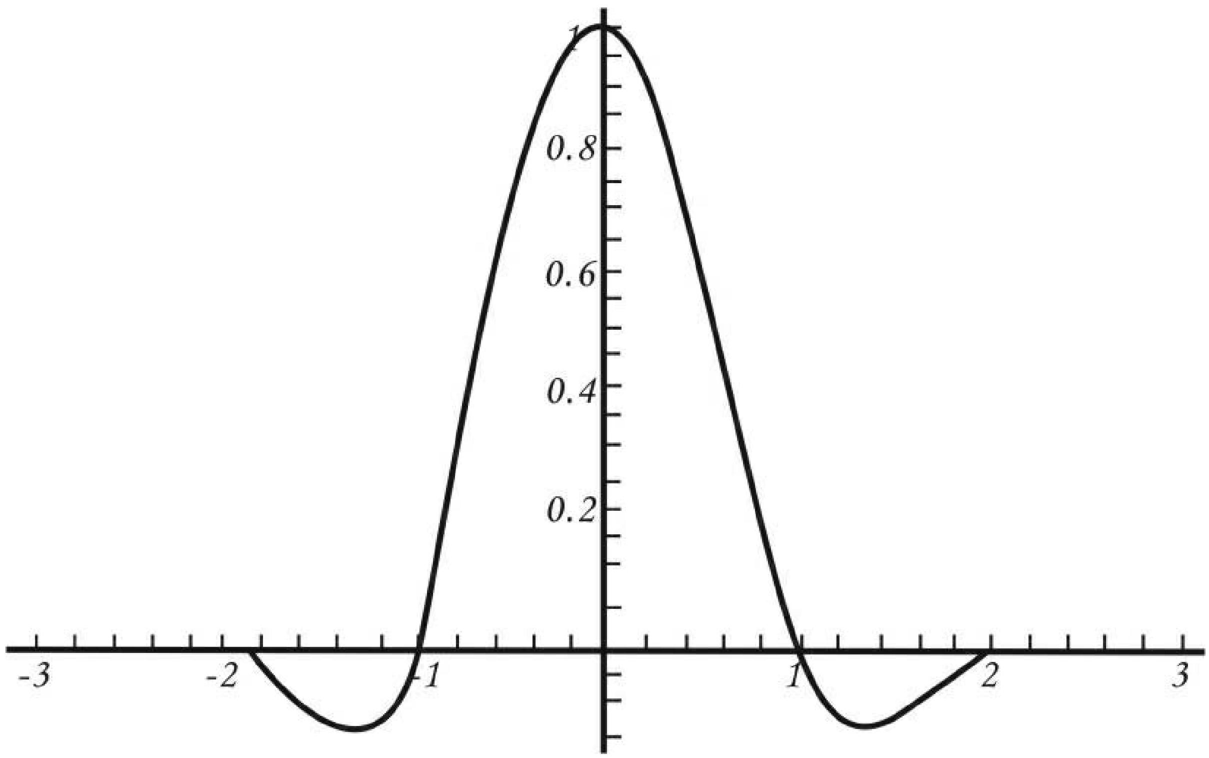
线性插值的假设是原图的灰度在两个像素之间是线性变化的，显然这是一种比较合理的假设。因此在一般情况下，双线性插值都能取得不错的效果。更精确的方法是采用曲线插值，即认为像素之间的灰度变化规律符合某种曲线方程，当然这种处理的计算量是很大的。

4.7.3 高阶插值

在几何运算的一些情况中，双线性插值的平滑作用会使图像的细节退化，而其斜率的不连续性则会导致变换产生不希望的结果。这些都可以通过高阶插值得到弥补，高阶插值常用卷积来实现。输出像素的值为输入图像中距离它最近的4×4领域内采样点像素值的加权平均值。

下面以三次插值为例，它使用了如下的三次多项式来逼近理论上的最佳插值函数 sin(x)/x，如图4.15所示。

$$S(x)=\begin{cases}1-2|x|^2+|x|^3 & 0\leqslant|x|<1\\ 4-8|x|+5|x|^2-|x|^3 & 1\leqslant|x|<2\\ 0 & |x|\geqslant 2\end{cases}$$



▲图4.15 高阶插值示意图

上式中|x|是周围像素沿x方向与原点的距离。待求像素（x,y）的灰度值由其周围 16个点的灰度值加权插值得到。计算公式如下。

$$f(x,\ y)=f(i+u,j+v)=ABC$$

式中：

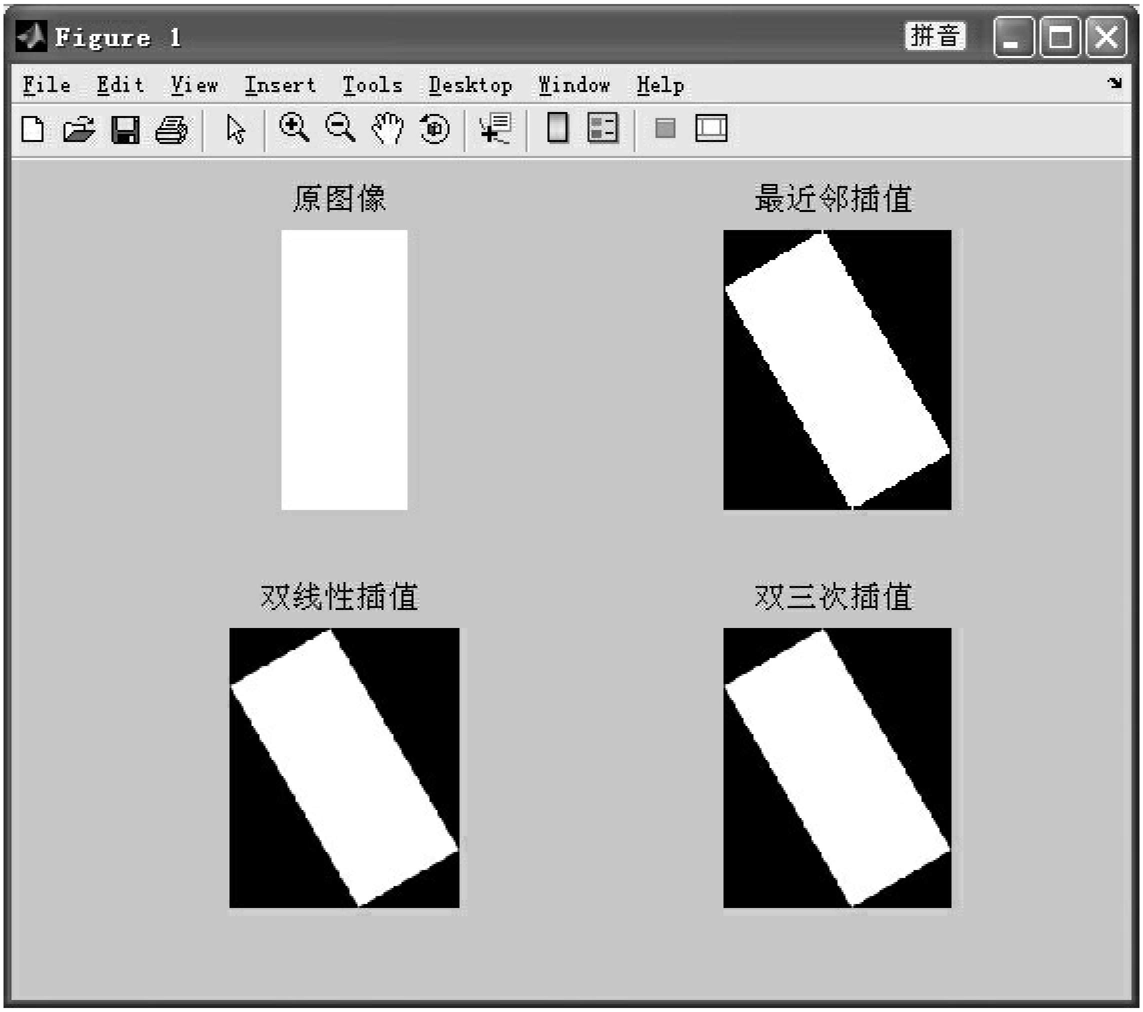
$$A=\begin{pmatrix} S(1+v) \\ S(v) \\ S(1-v) \\ S(2-v) \end{pmatrix}^T ;\quad C=\begin{pmatrix} S(1+u) \\ S(u) \\ S(1-u) \\ S(2-u) \end{pmatrix} ;$$

$$B=\begin{pmatrix} f(i-1,j-1) & f(i-1,j) & f(i-1,j+1) & f(i-1,j+2) \\ f(i,j-1) & f(i,j) & f(i,j+1) & f(i,j+2) \\ f(i+1,j-1) & f(i+1,j) & f(i+1,j+1) & f(i+1,j+2) \\ f(i+2,j-1) & f(i+2,j) & f(i+2,j+1) & f(i+2,j+2) \end{pmatrix}$$

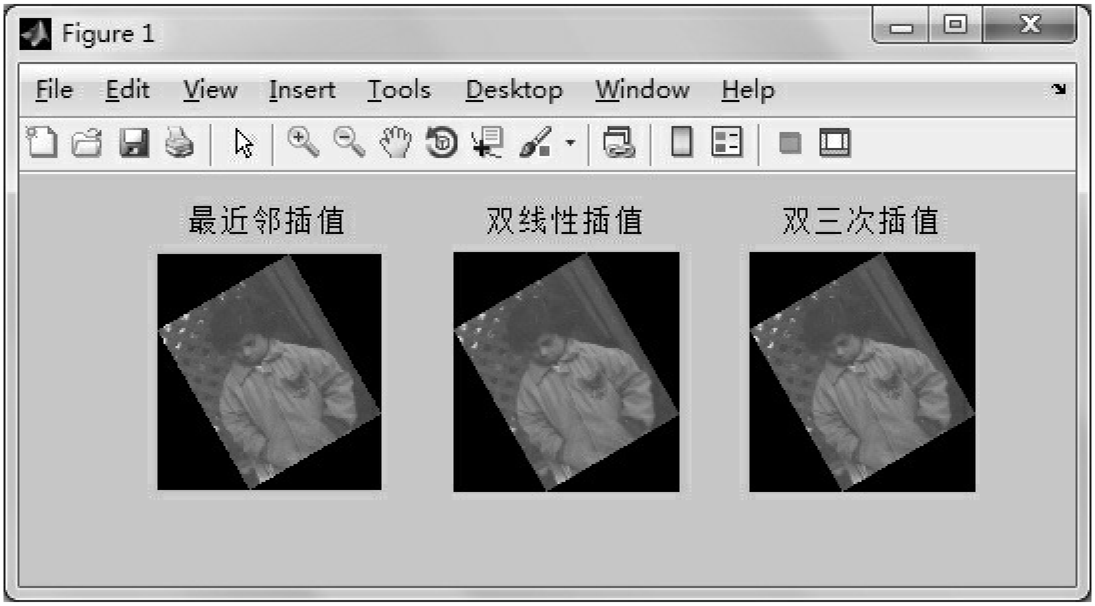
三次插值方法通常应用在光栅显示中，它在允许任意比例的缩放操作的同时，较好地保持了图像细节。

[例4.6]插值方法比较。

图4.16和图4.17分别给出了采用最近邻、双线性和三次插值时对于两幅不同图像的旋转效果。从图 4.17 可以看出最近邻的插值方法得到的结果还是可以接受的，但当图像中包含的像素之间灰度级有明显变化时（图4.16），从结果图像的锯齿形边可以看出三种插值方法的效果依次递减，最近插值的效果明显不如另外两个好，锯齿比较多，而双三次插值得出的图像较好地保持了图像的细节。这是因为参与计算输出点的像素值的拟合点个数不同，个数越多效果越精确，当然参与计算的像素个数会影响计算的复杂度。实验结果也清楚的表明：双三次插值法花费的时间比另外两种的要长一些。最近邻和线性插值的速度在此次图像处理中几乎分不出来。所以，在计算时间与质量之间有一个折中问题。



▲图4.16 插值方法比较效果图 1



▲图4.17 插值方法比较效果图 2

实现3种插值算法的Matlab代码如下所示。

```
% 插值算法比较
A=imread('../rectangle.bmp'); %读入本章的示例矩形图片
B = imrotate(A,30,'nearest');
C = imrotate(A,30,'bilinear');
D = imrotate(A,30,'bicubic');
%图像旋转30°的插值方法比较
subplot(2,2,1),imshow(A);
title('原图像');
subplot(1,3,1),imshow(B);
title('最近邻插值');
subplot(1,3,2),imshow(C);
title('双线性插值');
subplot(1,3,3),imshow(D);
title('双三次插值');
```

4.8 Matlab综合案例——人脸图像配准

在4.2～4.6节我们依据算法4.1从正变换的角度讲解了图像的平移、镜像、转置、缩放和旋转变换。而本节将要介绍的图像配准技术则是站在几何失真归一化的角度，以一种逆变换的思路来阐述几何变换。

4.8.1 什么是图像配准

所谓图像配准就是将同一场景的两幅或多幅图像进行对准。如航空照片的配准，以及在很多人脸自动分析系统中的人脸归一化，即要使各张照片中的人脸具有近似的大小，尽量处于相同的位置。

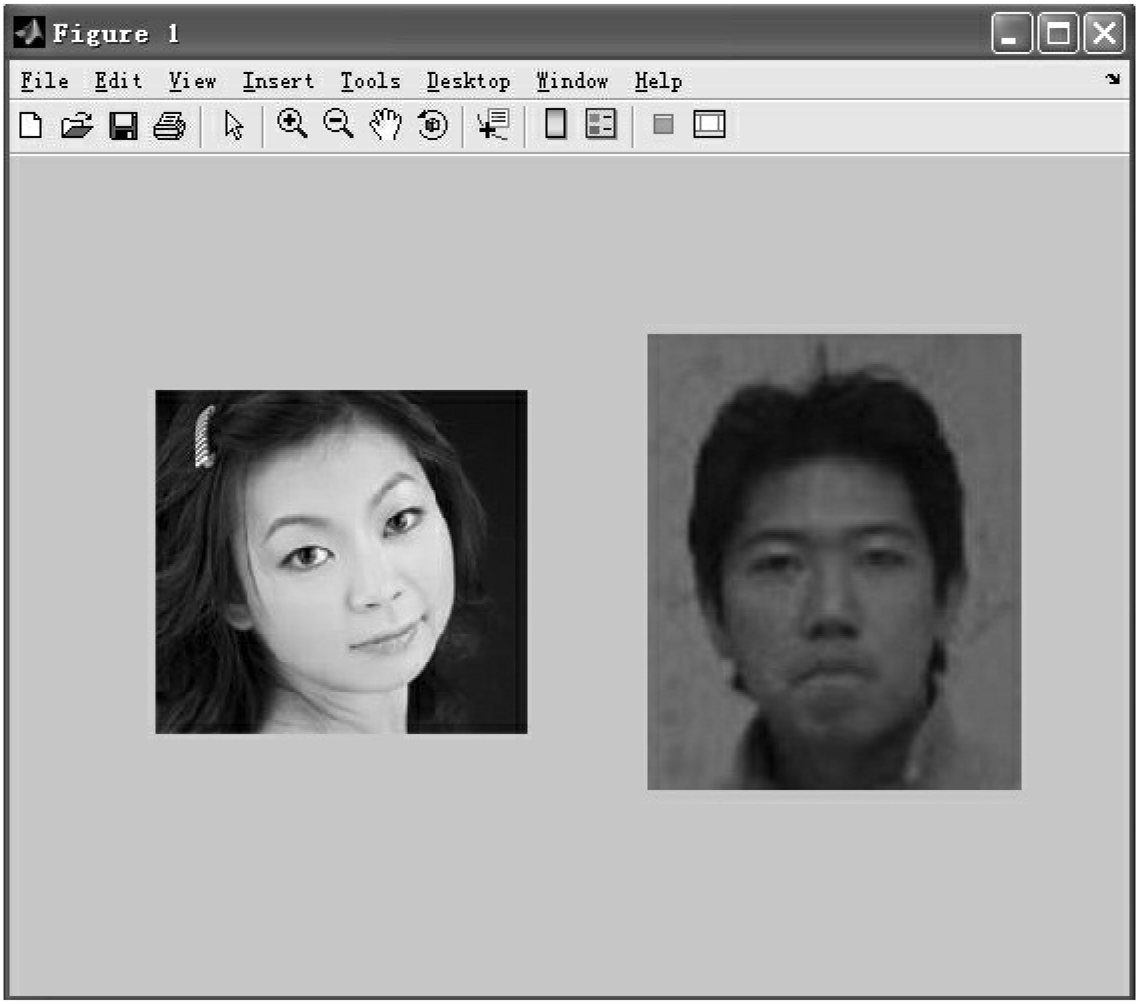
一般来说，我们以基准图像为参照，并通过一些基准点（fiducial points）找到适当的空间变换关系s和t，对输入图像进行相应的几何变换，从而实现它与基准图像在这些基准点位置上的对齐。

下面就以人脸图像的校准为例，学习如何在Matlab中实现图像配准。

4.8.2 人脸图像配准的Matlab实现

（1）读入基准图像和要配准的输入图像。配准前图像如图4.18所示。

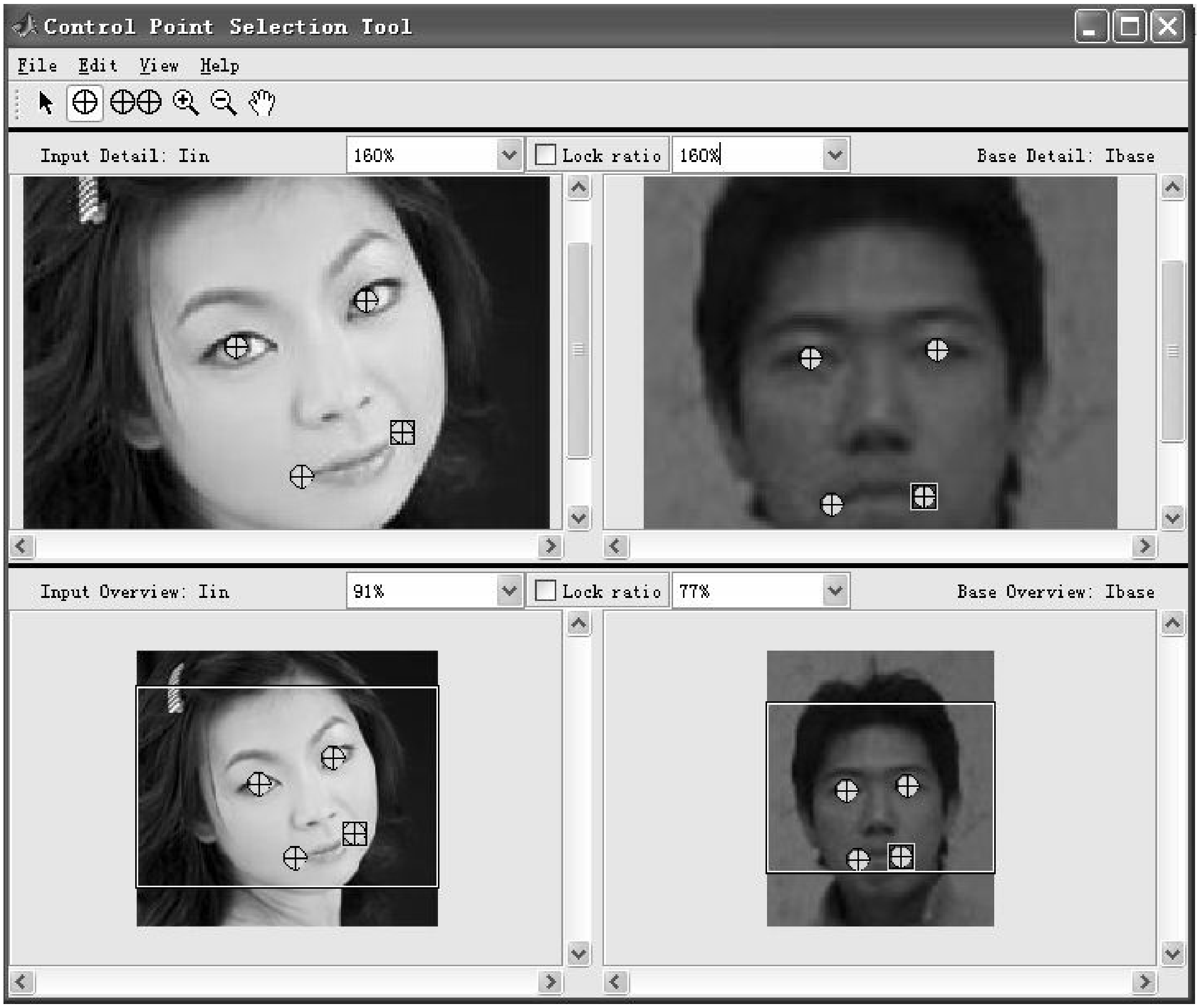
```
>> Iin = imread('face2.jpg');
>> Ibase = imread('face1.jpg');
>> figure
subplot(1, 2, 1), imshow(Iin);
subplot(1, 2, 2), imshow(Ibase);
```



▲图4.18 配准之前的图像，左侧为需要配准的图像，右侧为基准图像

（2）标注基准点对，并将其保存至工作空间。

利用 Matlab 提供的cpselect函数可以交互式地选择基准点。在命令行中按照下面的方式调用cpselect可以启动该交互工具。可以分别单击两幅图像中的相同部分选择成对儿的基准点，如眼睛和嘴角，如图4.19所示。



▲图4.19 利用 cpselect交互地选择基准点对

```
>> cpselect(Iin, Ibase);
>> input_points
input_points =
   81.8988 89.5000
  130.8988 72.0000
  106.3988 139.0000
  144.8988 122.0000
>> base_points
base_points =
```

64.1540 111.3750

112.1540 108.3750

72.1540 166.8750

107.1540 163.8750

注意调用时要将需要配准的图像作为第一个参数，将基准图像作为后一个参数。还有一点就是cpselect只接受灰度图像，如果需要处理RGB彩色图像，可以只给cpselect函数传递图像的一个层。

>>cpselect(Irgb(:,:,1), Ibase); % 只传递输入图像的红色分量图

单击交互工具File菜单下的Save Points to Workspace选项，可以将之前选择的基准点对保存至工作空间。默认情况下，输入图像中的基准点保存在变量input_points中，基准图像的基准点保存在变量base_points中。

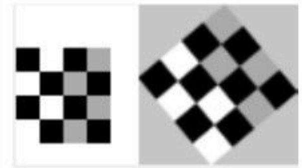
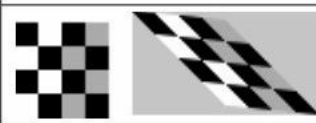
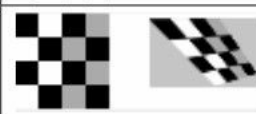
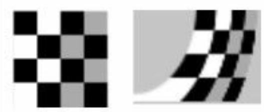
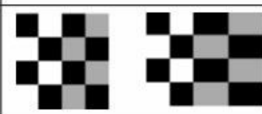
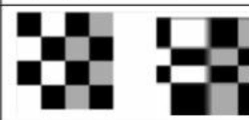
（3）指定要使用的变换类型。

根据之前得到的控制点对坐标input_points和base_points，利用cp2tform函数可以计算变换的参数。将基准点对作为输入传递给cp2tform，选择一种适当的变换类型，cp2tform函数就能够确定出该类型变换所需的参数。这实际上相当于一种数据拟合，cp2tform函数寻找能够拟合控制点对的变换参数，它返回一个TFORM结构的几何变换结构，其中就包括了几何变换的类型和参数。

>> tform = cp2tform(input_points, base_points, 'affine'); %仿射变换模型

参数'affine'表示选用投影变换。cpselect函数总共支持6种类型的变换，包括线性和非线性的，以及两种分段变换，如表4.2所示。实际应用中的关键是要结合失真的具体情况和变换需要选择合适的变换类型。

表4.2 变换类型及其说明

变换类型	适 用 情 况	最小控制点对	示　　例
linear conformal	当输入图像中的形状没有改变，但图像经过了平移、旋转以及比例缩放等变换后发生失真时使用本变换。变换后直线仍然是直线，平行线仍为平行的	2 对	
affine	当输入图像中的形状展示出错效果使用本变换。变换后直线仍然是直线，平行线仍为平行的，但矩形变成了平行四边形	3 对	
Projective	当场景显得倾斜时使用本变换。变换后直线仍然为直线，但平行线不再平行	4 对	
polynomial	当图像中的对象发生弯曲时使用本变换。拟合中选用的多项式的阶数越高，拟合效果越好，但配准后的图像比基准图像包含更多的曲线，同时也需要更多的基准点对	6 对（二阶） 10 对（3 阶） 15 对（4 阶）	
piecewise linear	当图像呈现出分段变形现象时使用本变换	4 对	
lwm	当图像中的变形具有局部化特点，并且分段线性条件不够充分时考虑使用本变换	6 对（推荐 12 对）	

（4）根据变换结构对输入图像进行变换，完成基于基准点的对准。

调用imtransform函数进行变换，从而实现配准，函数返回配准后的图像如图4.20（a）所示。

```
>> Iout = imtransform(Iin, tform);
>> figure
subplot(1, 2, 1), imshow(Iout);
subplot(1, 2, 2), imshow(Ibase);
```



（a）采用仿射模型



（b）采用投影模型

▲图4.20 配准后的图像

图4.20（b）给出了采用投影模型（调用cp2tform时第3个参数为'projective'）的配准效果，同图4.20（a）相比，其4个基准点的对齐效果更好，但输入图像的变形也更大一些。

第5章 空间域图像增强

图像增强是数字图像处理相对简单却最具艺术性的领域之一，增强的目的是消除噪声，显现那些被模糊了的细节或简单地突出一幅图像中我们感兴趣的特征。一个简单例子是增强图像的对比度，使其看起来更加一目了然。应记住，增强是图像处理中非常主观的领域，它以怎样构成好的增强效果这种人的主观偏好为基础，也正是这一点为其赋予了艺术性。这与图像复原技术刚好相反，图像复原也是改进图像外貌的一个处理领域，但它是客观的。

本章的知识和技术热点

● 空间域滤波的基础知识

● 相关和卷积

● 图像平滑，包括平均平滑和高斯平滑

● 中值滤波及其改进算法

● 图像锐化，包括梯度算子、拉普拉斯算子、高提升滤波和高斯-拉普拉斯变换

本章的典型案例分析

● 对椒盐噪声的平滑效果比较

● Laplacian与LoG算子的锐化效果比较

5.1 图像增强基础

5.1.1 为什么要进行图像增强

图像增强是指根据特定的需要突出一幅图像中的某些信息，同时削弱或去除某些不需要的信息的处理方法。其主要目的是使处理后的图像

对某种特定的应用来说，比原始图像更适用。因此，这类处理是为了某种应用目的而去改善图像质量的。处理的结果使图像更适合于人的观察或机器的识别系统。

应该明确的是增强处理并不能增强原始图像的信息，其结果只能增强对某种信息的辨别能力，而同时这种处理有可能损失一些其他信息。正因如此我们很难找到一个评价图像增强效果优劣的客观标准，也就没有特别通用模式化的图像增强方法，总是要我们根据具体期望的处理效果做出取舍。

5.1.2 图像增强的分类

图像增强技术基本上可分成两大类：一类是空间域增强，一类是频率域增强。本章着重介绍空间域增强技术，下一章讲述频率域图像增强。

空间域图像增强与频率域图像增强不是两种截然不同的图像增强技术，实际上在相当程度上说，它们是在不同的领域做同样的事情，是殊途同归的，只是有些滤波更适合在空间域完成，而有些则更适合在频率域中完成。

空间域图像增强技术主要包括直方图修正、灰度变换增强、图像平滑化以及图像锐化等。在增强过程中可以采用单一方法处理，但更多实际情况是需要采用几种方法联合处理，才能达到预期的增强效果（永远不要指望某个单一的图像处理方法可以解决全部问题）。

在第3章中通过灰度变换改善图像外观的方法，以及3.7、3.8节中的直方图灰度修正技术（即直方图均衡化和直方图规定化）都是图像增强的有效手段，这些方法的共同点是变换是直接针对像素灰度值的，与该像素所处的邻域无关，而空间域增强则是基于图像中每一个小范围（邻域）内的像素进行灰度变换运算，某个点变换之后的灰度由该点邻域之

内的那些点的灰度值共同决定，因此空间域增强也称为邻域运算或邻域滤波。空间域变换可使用下式描述。

$$g(x, y) = T[f(x, y)] \tag{5-1}$$

5.2 空间域滤波

滤波是信号处理中的一个概念，是将信号中特定波段频率滤除的操作，在数字信号处理中通常采用傅立叶变换及其逆变换实现。由于下面要学习的内容实际上和通过傅立叶变换实现的频域下的滤波是等效的，故而也称为滤波。空间域滤波主要直接基于邻域（空间域）对图像中像素执行计算，我们使用空间域滤波这一术语区别于第6章中将要讨论的频率域滤波。

5.2.1 空间域滤波和邻域处理

对图像中的每一点（x，y），重复下面的操作。

（1）对预先定义的以（x，y）为中心的邻域内的像素进行运算。

（2）将（2）中运算的结果作为（x，y）点新的响应。

上述过程就称为邻域处理或空间域滤波。一幅数字图像可以看成一个二维函数 f(x, y)，而 x-y平面表明了空间位置信息，称为空间域，基于x-y空间邻域的滤波操作叫作空间域滤波。如果对于邻域中的像素计算为线性运算，则又称为线性空间域滤波，否则称为非线性空间域滤波。

图 5.1 直观地展示了用一个3×3的模板（又称为滤波器、模板、掩模、核或者窗口）进行空间滤波的过程，模板w中，用黑笔圈出的是其中心。

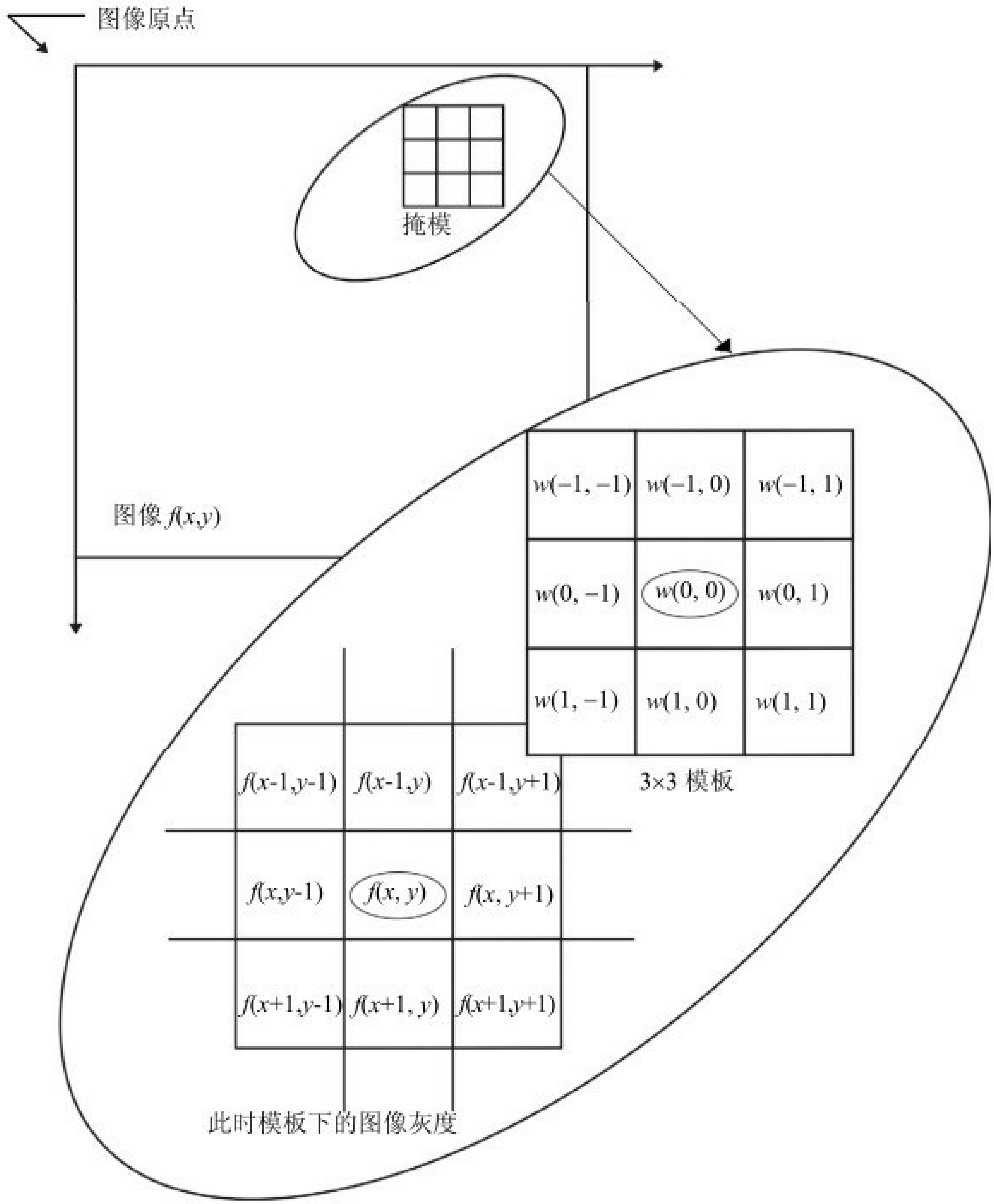
滤波过程就是在图像f(x，y)中逐点地移动模板，使模板中心和点

（x，y）重合，在每一点（x，y）处滤波器在该点的响应根据模板的具体内容并通过预先定义的关系来计算，一般来说模板中的非0元素指出了邻域处理的范围，只有那些当模板中心与点（x，y）重合时，图像f中和模板中非0像素重合的像素参与了决定点（x，y）像素值的操作，在线性空间滤波中模板的系数则给出了一种加权模式，即（x，y）处的响应由模板系数与模板下面区域的相应f的像素值的乘积之和给出。例如，对于图5.1而言，此刻对于模板的响应R为

$$R = w(-1, -1)f(x-1, y-1) + w(-1, 0)f(x-1, y) + \dots + w(0, 0)f(x, y) + \dots + w(1, 0)f(x+1, y) + w(1, 1)f(x+1, y+1)$$

更一般的情况，对于一个大小为m×n的模板，其中m=2a+1，n=2b+1，a、b均为正整数，即模板长与宽均为基数，且可能的最小尺寸为 3×3（偶数尺寸的模板由于其不具有对称性因而很少被使用，而1×1大小的模板的操作不考虑邻域信息，退化为图像点运算），可以将滤波操作形式化的表示为

$$g(x, y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s, t) f(x+s, y+t) \qquad (5\text{-}2)$$



▲图5.1 空间滤波示意

对于大小为M×N的图像 f(0..M−1, 0..N−1)，对 x = 0,1,2,…,M−1和 y = 0,1,2,…,N−1依次应用公式，从而完成了对于图像f所有像素的处理，

得到新的图像g。

5.2.2 边界处理

执行滤波操作要注意的一点是当模板位于图像边缘时，模板的某些元素很可能会位于图像之外的情况，这时需要对于在边缘附近执行滤波操作单独处理，以避免引用到本不属于图像的无意义的值（在Matlab中这将引起系统的警告，而在VC中很可能会由于非法访问内存而产生运行错误）。

以下3种策略都可以用来解决边界问题。

（1）收缩处理范围。处理时忽略位于图像f边界附近会引起问题的那些点，如对于图5.1中所使用的模板，处理时忽略图像 f四周一圈1个像素宽的边界，即只处理从 $x = 1,2,3,\ldots,M-2$和 $y = 1,2,3,\ldots N-2$（在Matlab中应为 $x = 2,3,4,\ldots,M-1$和 $y = 2,3,4,\ldots N-1$）范围内的点，从而确保了滤波过程中模板始终不会超出图像f的边界。

（2）使用常数填充图像。根据模板形状为图像f虚拟出边界，虚拟边界像素值为指定的常数，如 0，得到虚拟图像 f '。保证模板在移动过程中始终不会超出 f '的边界。

（3）使用复制像素的方法填充图像。和（2）基本相同，只是用来填充虚拟边界像素值的不是固定的常数，而是复制图像f本身边界的模式。

这些技巧在本章后面的小节程序设计实例中将给出具体实现。

5.2.3 相关和卷积

除了5.4.1小节给出的滤波过程外还有一种称为卷积的滤波过程。5.4.1中给出的滤波公式实际上是一种相关，而卷积的形式化表示略有不同。

$$g(x,y)=\sum_{s=-a}^{a}\sum_{t=-b}^{b}w(-s,-t)f(x+s,y+t) \quad (5\text{-}3)$$

尽管差别细微，但有本质不同，卷积时模板是相对其中心点做镜像后再对f位于模板下的子图像做加权和的，或者说在做加权和之前模板先要以其中心点为原点旋转 180°。如果忽略了这一细微差别将导致完全错误的结果，只有当模板本身是关于中心点对称时相关和卷积的结果才会相同。

5.2.4 滤波操作的Matlab实现

Matlab中与滤波相关的函数主要有imfilter和fspecial。Imfilter完成滤波操作，而fspecial可以为我们创建一些预定义的二维滤波器，直接供imfilter函数使用。

1．滤波函数imfilter的原型如下。

g = imfilter(f,w,option1,option2,...)

参数说明

● f是要进行滤波操作的图像。

● w是滤波操作所使用的模板，为一个二维数组。

● option1, option2, … 是可选项，具体可以包括以下选项。

（1）边界选项：主要针对5.2.2小节中提到的边界处理问题，如表5.1所示。

表5.1

合 法 值	含 义
X（X代表一个具体的数字）	用固定数值X填充虚拟边界，默认情况是用0填充
'symmetric'	填充虚拟边界的内容通过对靠近原图像边缘像素相对于原图像边缘做镜像得到
'replicate'	填充虚拟边界的内容总是重复与它最近的边缘像素
'circular'	认为原图像模式具有周期性，从而周期性地填充虚拟边界的内容

采用第一种方式固定值填充虚拟边界的问题是在边缘附近会产生梯度，采用后面三种方式填充可让边缘显得平滑。

（2）尺寸选项：由于滤波中填充了边界，有必要指定输出图像g的大小，如表5.2所示。

表5.2

合 法 值	含 义
'same'	输出图像 g 与输入图像 f 尺寸相同
'full'	输出图像 g 的尺寸为填充虚拟边界后的图象 f' 的尺寸，因而大于输入图像 f 的尺寸

（3）模式选项：指明滤波过程是相关还是卷积，如表5.3所示。

表5.3

合 法 值	含 义
'corr'	滤波过程为相关
'conv'	滤波过程为卷积

返回值

● g为滤波后的输出图像。

[例5.1]读入灰度图像f=cameraman.tif，用模板 $w = 1/9 * \begin{pmatrix} 111 \\ 111 \\ 111 \end{pmatrix}$ 对f进行相关滤波，采用重复的边界填充方式。

```
>> f = imread('../fatBaby.bmp'); %读入图像
>> imshow(f); %得到图5.2（a）的图像
>> w = [1 1 1; 1 1 1; 1 1 1] / 9 %滤波模板
w =
  0.1111    0.1111    0.1111
```

```
    0.1111    0.1111    0.1111
    0.1111    0.1111    0.1111
>> g = imfilter(f, w, 'corr', 'replicate'); %滤波
>> figure, imshow(g); %得到5.2（b）的图像
```

运行结果如图5.2所示。



（a） 滤波前图像 fatBaby.bmp　　（b） 图像（a）经滤波后的图像

▲图5.2 相关滤波前后对比

2．可创建预定义的二维滤波器的fspecial函数的常见调用格式为：

h = fspecial(type,parameters)

参数说明

● 参数type指定了滤波器的类型，其中一些类型的滤波器将在5.3和5.4中介绍，有些则将放到第11章的图像分割中，作为边缘检测的算子。type的一些合法值如表5.4所示。

表5.4

合 法 取 值	功 能 描 述
'average'	平均模板
'disk'	圆形邻域的平均模板
'gaussian'	高斯模板
'laplacian'	拉普拉斯模板
'log'	高斯-拉普拉斯模板
'prewitt'	Prewitt 水平边缘检测算子
'sobel'	Sobel 水平边缘检测算子

● 可选输入parameters是和所选定的滤波器类型type相关的配置参数，如尺寸和标准差等。如果不提供，则函数使用该类型的默认参数配置。

返回值

● 返回值h为特定的滤波器。

下面结合一些代表性的情况具体说明。

（1）h= fspecial('average',hsize)返回一个大小为hsize的平均模板滤波器h。参数hsize可以是一个含有2个分量的向量，指明h的行和列的数目；也可以仅为一个正整数，此时对应于模板为方阵的情况。hsize的默认值为[3 3]。

（2）h = fspecial('disk',radius)返回一个半径为 radius 的圆形平均模板，h 是一个(2radius+1)*(2radius+1)的方阵。半径 radius的默认值为5。

（3）h = fspecial('gaussian',hsize,sigma)返回一个大小为 hsize，标准差σ=sigma的高斯低通滤波器。hsize的默认值为[3 3]，sigma的默认值为0.5。

（4）h = fspecial('sobel')返回一个加强水平边缘的竖直梯度算子。

```
h= [ 1   2   1
     0   0   0
    −1  −2  −1 ]
```

如果需要检测竖直边缘，则使用h’。

5.3 图像平滑

图像平滑是一种可以减少和抑制图像噪声的实用数字图像处理技术。在空间域中一般可以采用邻域平均来达到平滑的目的。

5.3.1 平均模板及其实现

从图5.2滤波前后效果对比可以看出滤波后的图g有平滑或者说模糊的效果，这完全是模板w作用的结果。例5.1中的w提供了一种平均的加权模式，首先在以点（x，y）为中心3×3邻域内的点都参与了决定在新图像g中（x，y）点像素值的运算；而且所有系数都为1，表示它们在参与决定g（x，y）值的过程中贡献（权重）都相同；最后前面的系数是要保证整个模板元素和为1，这里应为1/9，这样就能让新图像同原始图像保持在一个灰度范围中（如[0，255]）。

这样的 w 叫做平均模板，是用于图像平滑的模板中的一种，相当于一种局部平均。更一般的平均模板为

$$w = \frac{1}{(2k+1)^2}\begin{pmatrix} 11\ldots1 \\ 11\ldots1 \\ . \\ . \\ . \\ 11\ldots1 \end{pmatrix}_{(2k+1)\times(2k+1)} \qquad （5\text{-}4）$$

1．工作原理

一般来说，图像具有局部连续性质，即相邻像素的数值相近，而噪

声的存在使得在噪声点处产生灰度跳跃，但一般我们可以合理地假设偶尔出现的噪声影响并没有改变图像局部连续的性质，例如下面的局部图像f_sub，灰色底纹标识的为噪声点，在图像中表现为亮区中的2个暗点。

f _sub=

200	215	212	208	196
198	5	202	199	221
199	207	202	201	211
203	218	210	210	198
203	218	210	0	198
200	215	212	208	205

对f用3×3的平均模板进行平滑滤波后，得到的平滑后图像g_sub为

g_sub =

181	184	186	206	205
180	182	183	206	207
181	183	184	206	208
206	208	186	182	181
207	210	189	183	180
206	209	189	184	181

显然，通过平滑滤波原局部图像f_sub中噪声点的灰度值得到了有效修正，像这样将每一个点用周围点的平均替代从而达到减少噪声影响的过程就称为平滑或模糊。

2．Matlab实现

利用imfilter和fspecial，并以不同尺寸的平均模板实现平均平滑的Matlab示例代码如例5.2所示。

[例5.2]不同尺寸平均模板的平均平滑效果。

```
I = imread('baby_noise.bmp');
>> figure, imshow(I) %得到图5.3（a）中的图像
>> h = fspecial('average', 3); % 3*3平均模板
>> I3 = imfilter(I, h, 'corr', 'replicate'); % 相关滤波，重复填充边界
>> figure, imshow(I3) %得到图5.3（b）中的图像
>> h = fspecial('average', 5) % 5*5平均模板
h =
   0.0400    0.0400    0.0400    0.0400    0.0400
   0.0400    0.0400    0.0400    0.0400    0.0400
   0.0400    0.0400    0.0400    0.0400    0.0400
   0.0400    0.0400    0.0400    0.0400    0.0400
   0.0400    0.0400    0.0400    0.0400    0.0400
>> I5 = imfilter(I, h, 'corr', 'replicate');
>> figure, imshow(I5) %得到图5.3（c）中的图像
>> h = fspecial('average', 7); % 7*7平均模板
>> I7 = imfilter(I, h, 'corr', 'replicate');
>> figure, imshow(I7) %得到图5.3（d）中的图像
```

上述程序的运行效果如图5.3所示，可以看出随着模板的增大，滤波过程在平滑掉更多的噪声的同时也使得图像变得越来越模糊，这是由平均模板的工作机理所决定的。当模板增大到7×7时，图像中的某些细节，如衣服上的褶皱已经难以辨识了，纽扣也变得相当模糊。实际上，当图像细节与滤波器模板大小相近时就会受到比较大的影响，尤其当它们的灰度值又比较接近时，混合效应导致的图像模糊会更明显。随着模板的进一步增大，像纽扣这样的细节都会被当作噪声平滑掉。因此，我们在确定模板尺寸时应考虑好要滤除的噪声点的大小，有针对性地进行滤波。



（a）受噪声污染的婴儿老照片 baby_noise.bmp



（b）图（a）经 3×3 的平均模板滤波



（c）图（a）经 5×5 的平均模板滤波



（d）图（a）经 7×7 的平均模板滤波

▲图5.3 不同大小的平均模板的平滑效果。

5.3.2 高斯平滑及其实现

1．理论基础

平均平滑对于邻域内的像素一视同仁，为了减少平滑处理中的模糊，得到更自然的平滑效果，很自然地想到适当加大模板中心点的权重，随着远离中心点，权重迅速减小，从而可以确保中心点看起来更接近于与它距离更近的点，基于这样的考虑得到的模板即为高斯模板。

常用的3×3的高斯模板如下所示。

$$w = 1/16 \times \begin{pmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{pmatrix} \tag{5-5}$$

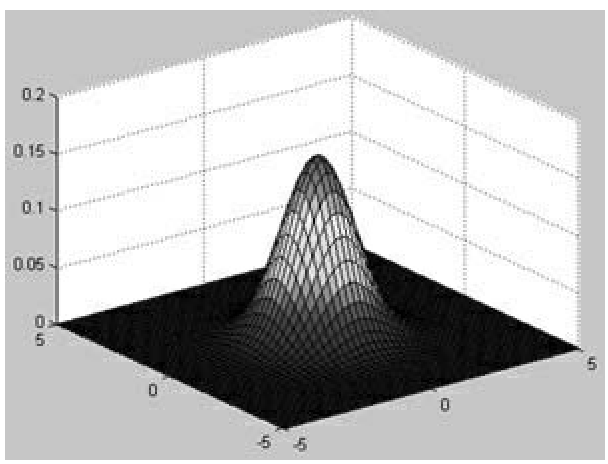
高斯模板名字的由来是二维高斯函数，即我们熟悉的二维正态分布密度函数，回忆一下，一个均值为0，方差为σ^2的二维高斯函数为

$$\varphi(x, y) = \frac{1}{2\pi\sigma^2} \exp(-\frac{(x^2 + y^2)}{2\sigma^2}) \tag{5-6}$$

图5.4给出了σ=1时二维高斯函数的三维示意图形。

高斯模板正是将连续的二维高斯函数的离散化表示，因此任意大小的高斯模板都可以通过建立一个（2k+1）×（2k+1）的矩阵M得到，其（i，j）位置的元素值可由下式确定。

$$M(i, j) = \frac{1}{2\pi\sigma^2} \exp(-\frac{((i-k-1)^2 + (j-k-1)^2)}{2\sigma^2}) \tag{5-7}$$



▲图5.4 二维高斯函数$\varphi(x, y) = \frac{1}{2\pi\sigma^2} \exp(-\frac{(x^2 + y^2)}{2\sigma^2})$取σ=1时的三维示意

● σ选择的小技巧

当标准差σ取不同的值时，二维高斯函数的形状会有很大的变化，因而在实际应用中选择合适的σ值非常重要：如果σ过小，偏离中心的所有像素权重将会非常小，相当于加权和响应基本不考虑邻域像素的作用，这样滤波操作退化为图像的点运算，无法起到平滑噪声的作用；相反如果σ过大，而邻域相对较小，这样在邻域内高斯模板将退化为平均模板；只有当σ取合适的值时才能得到一个像素值的较好估计。Matlab中σ的默认值为 0.5，在实际应用中，通常对 3×3 的模板取σ为0.8左右，对于更大的模板可以适当增大σ的值。

2．Matlab实现

采用不同的σ实现高斯平滑的Matlab代码如例5.3所示。

[例5.3] 不同的σ实现高斯平滑效果比较。

```
>> I = imread('baby_noise.bmp');
>> figure, imshow(I); %得到图5.5（a）中的图像
>>
>> h3_5 = fspecial('gaussian', 3, 0.5); % sigma=0.5的3*3高斯模板
>> I3_5 = imfilter(I, h3_5); % 高斯平滑
>> figure, imshow(I3_5); %得到图5.5（b）中的图像
>>
>> h3_8 = fspecial('gaussian', 3, 0.8); % sigma=0.8的3*3高斯模板
>> I3_8 = imfilter(I, h3_8);
>> figure, imshow(I3_8); %得到图5.5（c）中的图像
>>
>> h3_18 = fspecial('gaussian', 3, 1.8) % sigma=1.8的3*3高斯模板，接近于平均模板
h3_18 =
   0.0997    0.1163    0.0997
   0.1163    0.1358    0.1163
```

```
   0.0997$    0.1163    0.0997
>> I3_18 = imfilter(I, h3_18);
>> figure, imshow(I3_18); %得到图5.5（d）中的图像
>>
>> h5_8 = fspecial('gaussian', 5, 0.8);
>> I5_8 = imfilter(I, h5_8);
>> figure, imshow(I5_8); %得到图5.5（e）中的图像
>> imwrite(I5_8, 'baby5_8.bmp');
>>
>> h7_12 = fspecial(‘gaussian’, 7, 1.2);
>> I7_12 = imfilter(I, h7_12);
>> figure, imshow(I7_12); %得到图5.5（f）中的图像
>> imwrite(I7_12, 'baby7_12.bmp');
```

上述程序的运行结果如图5.5所示。图5.5（b）中的图像由于σ偏小而平滑效果不明显；当σ增大至1.8时，图5.5（d）中的高斯平滑效果类似于图5.3中图5.5（b）的平均平滑效果。随着模板的增大，原图中的噪声得到了更好的抑制，比较图5.5（f）和5.3图5.5（d）中的图像，我们注意到同样在大小为7×7的情况下，高斯滤波后的图像中图像细节被较好地保留。



（a）受噪声污染的婴儿老照片 baby_noise.bmp



（b）经 3×3，$\sigma = 0.5$ 的高斯模板滤波



（c）经 3×3，$\sigma = 0.8$ 的高斯模板滤波



（d）经 3×3，$\sigma = 0.8$ 的高斯模板滤波



（e）经 5×5，$\sigma = 0.8$ 的高斯模板滤波



（f）经 7×7，$\sigma = 1.2$ 的高斯模板滤波

▲图5.5 不同大小的高斯模板的平滑效果

上面介绍的平均平滑滤波器和高斯平滑滤波器都是线性平滑滤波器，在学习频率域滤波之后，还可以为它们赋予另外一个名字——低通滤波器。

5.3.3 自适应平滑滤波

利用平均模板的平滑在消除噪声的同时也使图像变得模糊，高斯平滑在一定程度上缓解了这些现象，但由平滑滤波机理可知这种模糊是不可避免的。这当然是我们所不希望的，于是想到选择性地进行平滑，即只在噪声局部区域进行平滑，而在无噪声局部区域不进行平滑，将模糊的影响降到最少，这就是自适应滤波的思想。

那么怎样判断该局部区域是包含噪声的需要平滑的区域还是无明显噪声的不需平滑的区域呢？这要基于噪声的性质来考虑，5.3.1 小节讨论了图像的局部连续性质，噪声的存在会使得在噪声点处产生灰度跳跃，从而使噪声点局部区域灰度跨度较大。因此可以选择如下 2 个标准中的 1个作为局部区域存在噪声的判据。

（1）局部区域最大值与最小值之差大于某一阈值T，即$\max(R)-\min(R)>T$，其中R代表该局部区域。

（2）局部区域方差大于某一阈值T，即$D(R)>T$，D(R)表示区域R中像素的方差。

自适应滤波算法的实现逻辑如算法5.1所示。

算法5.1

逐行扫描图像；

对每一个像素，以该像素作为中心，计算其周围区域R的统计特征，如最大值、最小值和方差等；

如果区域R的特征满足选定的噪声判据

根据选定的模板计算邻域加权和作为该点的响应；

否则

不处理该点；

对于那些噪声位置具有随机性和局部性的图像，自适应的滤波具有非常好的效果。有兴趣的读者可自己编制程序实现自适应的高斯平滑算法，应用于具有上述特点的噪声图像中，并和我们给出的标准高斯平滑效果进行比较。

5.4 中值滤波

中值滤波本质上是一种统计排序滤波器。对于原图像中某点（i, j），中值滤波以该点为中心的邻域内的所有像素的统计排序中值作为（i，j）点的响应。

中值不同于均值，是指排序队列中位于中间位置的元素的值，例如，采用3×3中值滤波器，某点（i，j）的 8 个邻域的一系列像素值为12、18、18、11、23、22、13、25、118，统计排序结果为11、12、13、18、18、22、23、25、118。排在中间位置（第5位）的18即作为（i，j）点中值滤波的响应g(i，j)。显然，中值滤波并非线性滤波器。

5.4.1 性能比较

中值滤波对于某些类型的随机噪声具有非常理想的降噪能力，对于线性平滑滤波而言，在处理的像素邻域之内包含噪声点时，噪声的存在总会或多或少地影响该点的像素值的计算，（对于高斯平滑影响程度同噪声点到中心点的距离成正比），但在中值滤波中噪声点则常常是直接被忽略掉的；而且同线性平滑滤波器相比，中值滤波在降噪同时引起的模糊效应较低。中值滤波的一种典型应用是消除椒盐噪声。

下面首先简单介绍一下常见的噪声模型，接着给出中值滤波的

Matlab实现。

1．噪声模型

Matlab中为图片加噪声的语句是

J = imnoise(I,type,parameters);

参数说明

● I为原图像。

● 可选参数type指定了噪声类型，常用的噪声类型如表5.5所示。

表5.5

合法取值	功能描述
'gaussian'	高斯白噪声：如果一个噪声，它的幅度分布服从高斯分布，则称之为高斯噪声。而如果它的功率谱密度（功率谱的概念见第 6 章）又是均匀分布的，则称它为高斯白噪声
'salt & pepper'	椒盐噪声因其在图像中的表现形式而得名，如图 5.6（b）是对 5.6（a）中的图像添加了椒盐噪声后的效果，黑点如同胡椒，白点好似盐粒。椒盐噪声是由图像传感器，传输信道，解码处理等产生的黑白相间的亮暗点噪声。椒盐噪声往往由图像切割引起

返回值

● J为添加了噪声后的图像。

提示

使用imnoise（’gaussian’, m, v）添加高斯噪声时，相当于对原图像中每一个像素叠加一个从均值为m、方差为v的高斯分布产生的随机样本值。当m=0时，较小的方差 v 通常保证了高斯分布有一个较大的概率产生值在0附近的随机样本（高斯分布密度函数f(x)在x=0附近具有最大值），从而在大部分像素位置对原图像影响较小。

2．中值滤波的**Matlab**实现

Matlab提供了medfilt2函数实现中值滤波，原型为

I2 = medfilt2(I1, [m,n]);

参数说明

● I1是原图矩阵。

● m和n是中值滤波处理的模板大小，默认3×3。

返回值

● 输出I2是中值滤波后的图像矩阵。

[例5.4] 对椒盐噪声的平滑效果比较

下面的程序分别给出了对于一幅受椒盐噪声污染的图像，平均平滑、高斯平滑和中值滤波的处理效果。

```
>> I = imread('lena_salt.bmp');
>> imshow(I); %得到图5.6（a）的图像
>> J=imnoise(I,'salt & pepper');%为图像叠加椒盐噪声
>> figure, imshow(J); %得到图5.6（b）的图像
>> w = [1 2 1;
      2 4 2;
      1 2 1] / 16;
>> J1=imfilter(J, w, 'corr', 'replicate'); %高斯平滑
>> figure, imshow(J1); %得到图5.6（c）的图像
>> w = [1 1 1;
       1 1 1;
       1 1 1] / 9;
>> J2=imfilter(J, w, 'corr', 'replicate');%平均平滑
>> figure, imshow(J2); %得到图5.6（d）的图像
>> J3=medfilt2(J,[3,3]);%中值滤波
>> figure, imshow(J3); %得到图5.6（e）的图像
```

程序运行结果如图5.6所示，从中可见线性平滑滤波在降噪的同时不可避免地造成了模糊，而中值滤波在有效抑制椒盐噪声同时模糊效应明显低得多，因而对于椒盐噪声污染的图像，中值滤波要远远优于线性平滑滤波。



（a）原图像 lena_salt.bmp



（b）椒盐噪声污染的图像



（c）3×3 高斯平滑效果



（d）3×3 平均平滑效果



（e）3×3 中值滤波效果



（f）改进的中值滤波效果

▲图5.6 几种滤波器对于椒盐噪声污染图像的性能比较

5.4.2 一种改进的中值滤波策略

中值滤波效果依赖于滤波窗口的大小，太大会使边缘模糊，太小了则去噪效果不好。因为噪声点和边缘点同样是灰度变化较为剧烈的像素，普通中值滤波在改变噪声点灰度值的时候，会一定程度地改变边缘像素灰度值。但是噪声点几乎都是邻域像素的极值，而边缘往往不是，因此可以利用这个特性来限制中值滤波。

具体的改进方法如下。逐行扫描图像，当处理每一个像素时，判断该像素是否是滤波窗口所覆盖下邻域像素的极大或者极小值。如果是，则采用正常的中值滤波处理该像素；如果不是，则不予处理。在实践中这种方法能够非常有效地去除突发噪声点，尤其是椒盐噪声，而几乎不影响边缘。

由于算法可以根据局部邻域的具体情况而自行选择执行不同的操作，因此改进的中值滤波也称为自适应中值滤波。这里将编制改进中值滤波程序的任务作为练习留给读者。

改进后中值滤波的处理效果如图5.6（f）所示。对比图5.6（f）和图5.6（e），我们不难发现，图5.6（f）在完美地滤除了椒盐噪声的同时，在图像细节（如帽子的褶皱）上，较图5.6（e）有了更好的保留，其他边缘也更加清晰，基本和原图一致。

5.4.3 中值滤波的工作原理

与线性平滑滤波考虑邻域中每个像素的作用不同，中值滤波在每个n×n邻域内都会忽略掉那些相对于邻域内大部分其余像素更亮或更暗，并且所占区域小于像素总数一半（$n^2/2$）的像素的影响，而实际上满足

这样条件被忽略掉的像素往往就是噪声。

注意

作为一种非线性滤波，中值滤波有可能会改变图像的性质，因而一般不适用于像军事图像处理、医学图像处理等领域。

5.5 图像锐化

图像锐化的目的是使模糊的图像变得更加清晰起来。其应用广泛，从医学成像到工业检测和军事系统的制导等。

5.5.1 理论基础

图像锐化主要用于增强图像的灰度跳变部分，这一点与图像平滑对灰度跳变的抑制正好相反，事实上从平滑与锐化的两种运算算子上也能说明这一点，线性平滑都是基于对图像邻域的加权求和或者说积分运算的，而锐化则通过其逆运算导数（梯度）或者说有限差分来实现。

在讨论平滑的时候我们提到了噪声和边缘都会使图像产生灰度跳变，为了在平滑时能够将噪声和边缘区别对待，还在5.3.5中给出了一种自适应滤波的解决方案。同样的，在锐化处理中如何区分开噪声和边缘仍然是我们要面临的一个课题，只是在平滑中要平滑的是噪声，希望处理不要涉及边缘；而在锐化中要锐化的对象是边缘，希望处理不要涉及噪声。

5.5.2 基于一阶导数的图像增强——梯度算子

回忆一下高等数学中梯度的定义，对于连续二维函数f(x，y)，其在点（x，y）处的梯度是下面的二维列向量。

$$\nabla f = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

式中：$\frac{\partial f}{\partial x} = \lim_{\varepsilon \to 0} \frac{f(x+\varepsilon, y) - f(x, y)}{\varepsilon}$，为在点（x，y）处f对x的偏导；

$\frac{\partial f}{\partial y} = \lim_{y \to 0} \frac{f(x, y+\varepsilon) - f(x, y)}{\varepsilon}$，为在点（x，y）处f对y的偏导；

梯度的方向就是函数f(x，y)最大变化率的方向。

梯度的幅值作为变化率大小的度量，其值为

$|\nabla f(x,y)| = \sqrt{(\frac{\partial f}{\partial x})^2 + (\frac{\partial f}{\partial y})^2}$。

对于离散二维离散函数f（i，j），可以用有限差分作为梯度幅值的一个近似。

$$|\nabla f(i,j)| = \sqrt{[f(i+1,j) - f(i,j)]^2 + [f(i,j+1) - f(i,j)]^2}$$

尽管梯度幅值和梯度两者之间有着本质的区别，但在数字图像处理中提到梯度时，往往不加区分，即将上式的梯度幅值称为梯度。

上式中包括平方和开方，不方便计算，因此可近似为绝对值的形式。

$$|\nabla f(i,j)| = |f(i+1,j) - f(i,j)| + |f(i,j+1) - f(i,j)| \qquad （5\text{-}8）$$

而在实际使用中，经常被采用的是另外一种近似梯度—Robert交叉

梯度。

$$|\nabla f(i,j)| = |f(i+1,j+1) - f(i,j)| + |f(i,j+1) - f(i+1,j)| \quad (5\text{-}9)$$

1．Robert交叉梯度

Robert交叉梯度对应的模板为

$$w1 = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \qquad w2 = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$$

式中：w1对接近+45°边缘有较强响应；w2对接近−45°边缘有较强响应。

[例5.5] 基于Robert交叉梯度的图像锐化

有了前面学习的滤波的知识，只要分别以w1和w2为模板，对原图像（a）进行滤波就可得到G1和G2，而根据公式5-9最终的Robert交叉梯度图像（b）为

$$G = |G1| + |G2|$$

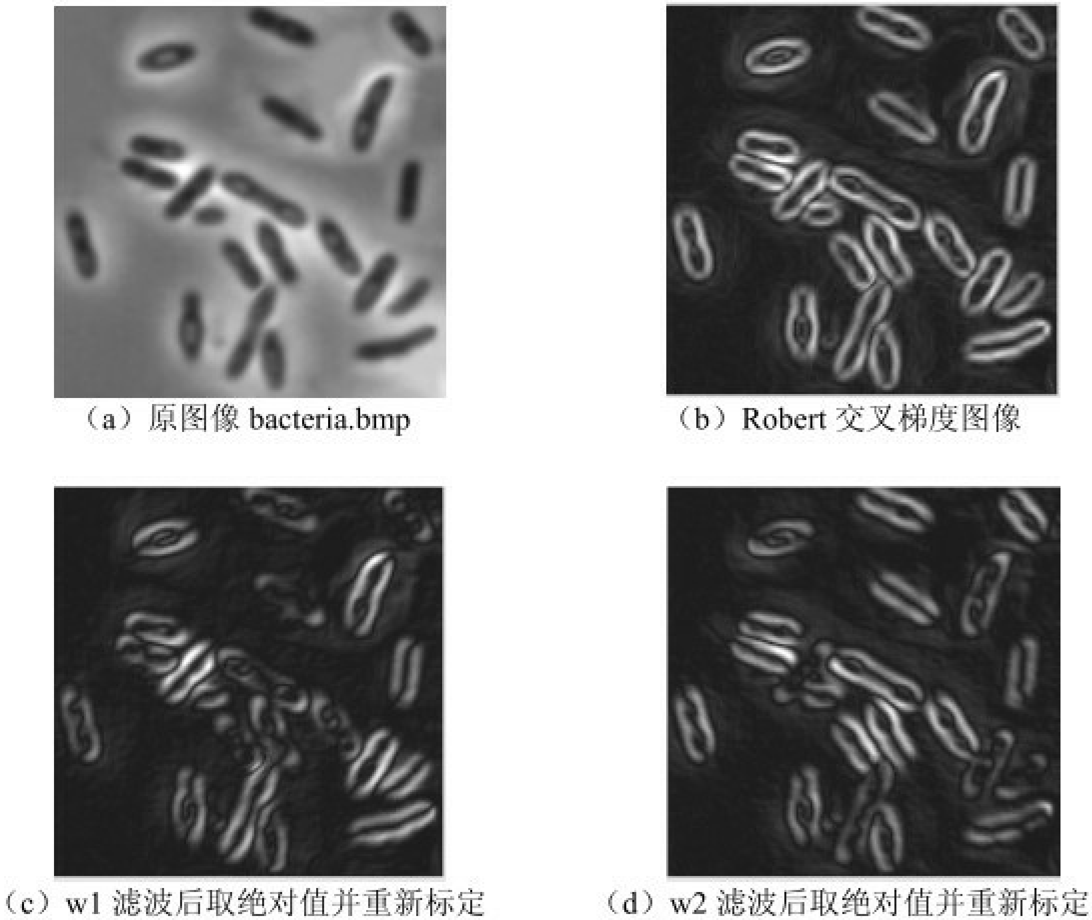
在进行锐化滤波之前，我们要将图像类型从uint8转换为double，这是因为锐化模板的负系数常常使得输出产生负值，如果采用无符号的uint8型，则负值会被截断。

在调用函数imfilter时，还要注意不要使用默认的填充方式，因为Matlab默认会在滤波时进行“0”填充，这会导致图像在边界处产生一个人为的灰度跳变，从而在梯度图像中产生高响应，而这些人为高响应值的存在将导致对图像中真正的边缘和其他我们关心的细节的响应在输出梯度图像中被压缩在一个很窄的灰度范围，同时也影响显示的效果。我们这里采用了replicate的重复填充方式，也可采用symmetric的对称填充方式。

程序实现如下。

```
>> I = imread('bacteria.bmp');
>> imshow(I); % 显示图5.7(a)
>> I = double(I); % 转换为double型，这样可以保存负值，否则uint8型会把负值截掉
>> w1 = [-1 0; 0 1]
w1 =
   -1   0
   0   1
>> w2 = [0 -1; 1 0]
w2 =
   0   -1
   1   0
>> G1 = imfilter(I, w1, 'corr', 'replicate'); % 以重复方式填充边界
>> G2 = imfilter(I, w2, 'corr', 'replicate');
>> G = abs(G1) + abs(G2); % 计算Robert梯度
>> figure, imshow(G, []); % 显示图5.7(b)
>> figure, imshow(abs(G1), []); % 显示图5.7(c)
>> figure, imshow(abs(G2), []); % 显示图5.7(d)
```

上述程序的运行结果如图5.7所示。由于G1和G2中都可能有负值，图5.7（c）和图5.7（d）分别是对G1和G2取绝对值后的图像，图5.7（c）中接近+45°边缘较明显，而图5.7（d）中则突显出接近−45°方向的边缘，这与直接分析w1和w2模板结构得出的结论是一致的。



▲图5.7 Robert交叉梯度锐化

提示

为便于观察效果，图 5.7（b）、图 5.7（c）、图 5.7（d）都做了显示时的重新标定，即将图像的灰度范围线性变换到0～255，并使得图像的最小灰度值为0，最大灰度值为 255。在Matlab中只需在用 imshow函数显示图像时加一个参数 [ ]即可。

2．Sobel梯度

由于滤波时我们总是喜欢奇数尺寸的模板，因而一种计算Sobel梯度的Sobel模板更加常用。

$$w1 = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

对水平边缘有较大响应的竖直梯度

$$w2 = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

对竖直边缘有较大响应的水平梯度

例5.6的Matlab程序计算了一幅图像的竖直和水平梯度，它们的和可以作为完整的Sobel梯度。

[例5.6] 基于Sobel梯度的图像锐化

```
I = imread('bacteria.bmp'); %读入原图像
>> w1 = fspecial('sobel'); %得到水平sobel模板
>> w1
w1 =
   1    2    1
   0    0    0
   -1    -2    -1
>> w2 = w1' %转置得到竖直soble模板
w2 =
   1    0    -1
   2    0    -2
   1    0    -1
>> G1 = imfilter(I, w1); %水平Sobel梯度
```

>> G2 = imfilter(I, w2); %竖直Sobel梯度

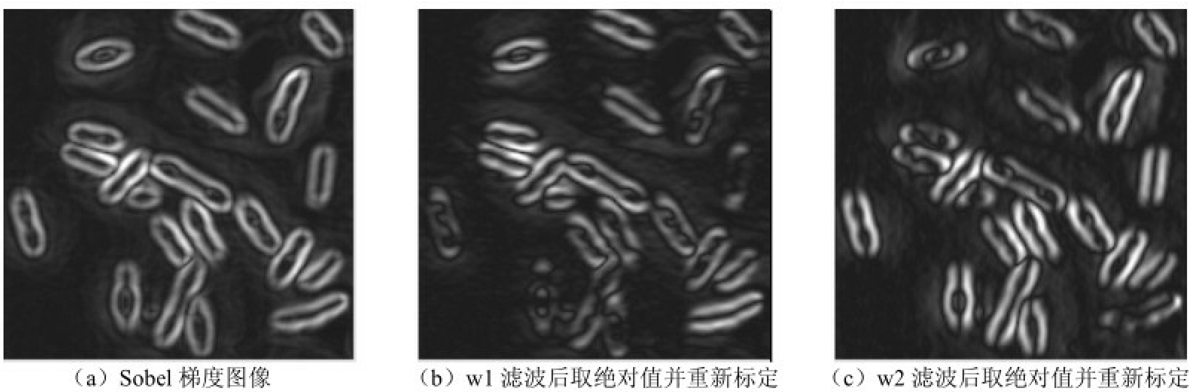
>> G = abs(G1) + abs(G2); %Sobel梯度

>> figure, imshow(G1, []) %得到图5.8（a）

>> figure, imshow(G2, []) %得到图5.8（b）

>> figure, imshow(G, []) %得到图5.8（c）

上述程序运行后，图5.7（a）所示的Sobel梯度锐化效果如图5.8所示。



▲图5.8 Sobel梯度锐化效果，（b）中接近水平方向的边缘较明显，（c）中接近竖直方向的边缘较明显

还可以直接利用Matlab梯度函数gradient计算Sobel梯度，程序如下。

>> I = imread('bacteria.bmp');

>> imshow(I);

>> I = double(I); % 计算梯度之前要转换为double

>> [Gx Gy] = gradient(I); % 计算x,y方向梯度

>> G = abs(Gx) + abs(Gy); %计算整体梯度

>> figure, imshow(G);

>> figure, imshow(G, []); % 整体梯度图像

>> figure, imshow(Gx, []); % x方向梯度图像（突显偏竖直方向的边缘）

>> figure, imshow(Gy, []); % y方向梯度图像（突显偏水平方向的边缘）

关于梯度算子更详细的讨论留在第11章边缘检测与图像分割中进行。

5.5.3 基于二阶微分的图像增强——拉普拉斯算子

下面介绍一种对于图像锐化而言应用更为广泛的基于二阶微分的拉普拉斯（Laplacian）算子。

1．理论基础

二维函数f(x, y)的二阶微分（拉普拉斯算子）定义为

$$\nabla^2 f(x,y) = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

对于离散的二维图像f(i, j)，可以用下式作为对二阶偏微分的近似。

$$\frac{\partial^2 f}{\partial x^2} = (f(i+1,j)-f(i,j))-(f(i,j)-f(i-1,j)) = f(i+1,j)+f(i-1,j)-2f(i,j)$$

$$\frac{\partial^2 f}{\partial y^2} = (f(i,j+1)-f(i,j))-(f(i,j)-f(i,j-1)) = f(i,j+1)+f(i,j-1)-2f(i,j)$$

将上面两式相加就得到用于图像锐化的拉普拉斯算子。

$$\nabla^2 f = [f(i+1,j)+f(i-1,j)+f(i,j+1)+f(i,j-1)]-4f(i,j) \qquad (5\text{-}10)$$

对应的滤波模板如下。

$$W1 = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

因为在锐化增强中，绝对值相同的正值和负值实际上表示相同的响应，故也等同于使用如下的模板W2。

$$W2 = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

分析拉普拉斯模板的结构，可知这种模板对于90°的旋转是各向同性的，所谓对于某角度各向同性是指把原图像旋转该角度后再进行滤波与先对原图像滤波再旋转该角度的结果相同。这说明拉普拉斯算子对于接近水平和接近竖直方向的边缘都有很好的增强，从而也就避免我们在使用梯度算子时要进行两次滤波的麻烦。更进一步，我们还可以得到如下对于45°旋转各向同性的滤波器。

$$W3 = \begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix} \text{ 和 } W4 = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

沿用高斯平滑模板的思想，根据到中心点的距离给模板周边的点赋予不同的权重，还可得到如下的模板W5。

$$W5 = \begin{bmatrix} 1 & 4 & 1 \\ 4 & -20 & 4 \\ 1 & 4 & 1 \end{bmatrix}$$

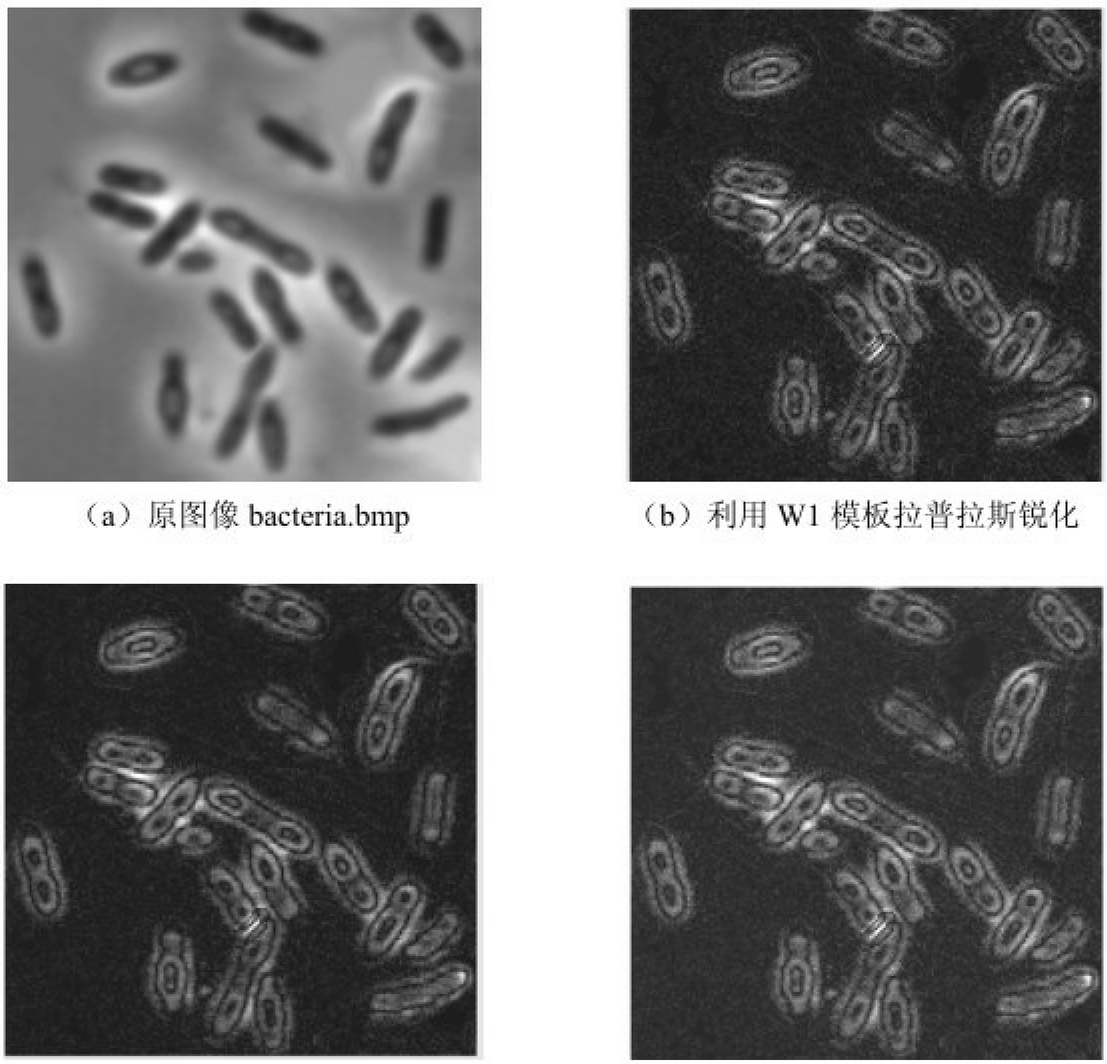
2．Matlab实现

分别使用上述的3种拉普拉斯模板的Matlab滤波程序如例5.7所示。

[例5.7] 基于3种拉普拉斯模板的滤波

```
>> I = imread('bacteria.bmp');
>> figure, imshow(I); %得到图5.9（a）
>> I = double(I);
>> w1 = [0 -1 0; -1 4 -1; 0 -1 0]
w1 =
   0   -1    0
   -1   4   -1
    0   -1    0
>> L1 = imfilter(I, w1, 'corr', 'replicate');
>> w2 = [-1 -1 -1; -1 8 -1; -1 -1 -1]
w2 =
   -1   -1   -1
   -1    8   -1
   -1   -1   -1
>> L2 = imfilter(I, w2, 'corr', 'replicate');
>> figure, imshow(abs(L1), []);%得到图5.9（b）
>> figure, imshow(abs(L2), []);%得到图5.9（c）
>> w3 = [1 4 1; 4 -20 4; 1 4 1]
w3 =
   1    4    1
   4   -20   4
   1    4    1
>> L3 = imfilter(I, w3, 'corr', 'replicate');
>> figure, imshow(abs(L3), []);%得到图5.9（d）
```

上述程序运行结果如图5.9所示。对于细菌图像拉普拉斯锐化效果与之前的Robert与Sobel梯度锐化明显不同的一点是输出图像中的双边缘。此外，我们还注意到拉普拉斯锐化似乎对一些离散点有较强的响应。当然由于噪声也是离散点，因此这个性质有时是我们所不希望的。



（a）原图像 bacteria.bmp　（b）利用 W1 模板拉普拉斯锐化

（c）利用 W3 模板拉普拉斯锐化　（d）利用 W5 模板拉普拉斯锐化

▲图5.9 拉普拉斯锐化效果

5.5.4 基于一阶与二阶导数的锐化算子的比较

5.5.2和5.5.3小节中我们分别介绍了基于一阶导数的Robert和Sobel算

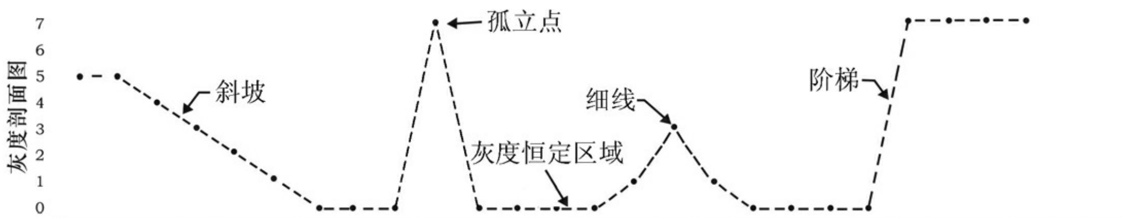
子，以及基于二阶导数的Lapalacian算子，并且通过图5.8～5.9从直观上观察和比较了它们的处理效果。下面将进行更为精确的分析和比较，以找到一些能够在实践中具有指导意义的一般性规律。

设图 5.10 最上面部分的灰度剖面图对应于图像中的一条具有代表性的水平像素线，其中包括了灰度较缓变化的斜坡（软边缘）、孤立点（很可能为噪声）、细线（细节）以及灰度跳变的阶梯（硬边缘）。为了简单起见，考虑图像中只有8个灰度级的情况。图5.10中下面的一行给出了这条像素线中各个像素的灰度值，由此计算出的一阶微分和二阶微分在图中的第三行和第四行中给出。由于这里的像素线在图像中是水平分布的，因此式 5-9 和式 5-10 可简化为一维的形式，即一维情况下的一阶微分为

$$\frac{\partial f}{\partial x} = f(x+1) - f(x) \qquad （5\text{-}11）$$

一维情况下的二阶微分为

$$\frac{\partial^2 f}{\partial x^2} = f(x+1) + f(x-1) - 2f(x) \qquad （5\text{-}12）$$



灰度值	5	5	4	3	2	1	0	0	0	7	0	0	0	0	1	3	1	0	0	0	0	7	7	7	7	·	·
一阶微分	-1	-1	-1	-1	-1	0	0	7	-7	0	0	0	1	2	-2	-1	0	0	0	7	0	0	0				
二阶微分	-1	0	0	0	0	1	0	7	-14	6	0	0	1	1	-4	1	1	0	0	7	-7	0	0				

▲图5.10 一阶与二阶微分比较

通过分析这个典型的灰度变化模型，我们就可以很好地比较噪声点、细节以及边缘的一阶和二阶微分结果。

首先注意到沿着整个斜坡（软边缘），一阶微分都具有非0响应，并且当这种斜坡的灰度过渡近似线时，对应于变化率的一阶微分的响应为恒定值（这里为−1）；而二阶微分的非0响应则只出现在斜坡的起始和终点处，在灰度变化率恒定的斜面上二阶微分值为0，这就是图5.9中的拉普拉斯锐化图像周围出现双边缘的原因。由此得出结论，对于图像中的软边缘，一阶微分通常产生较粗的边缘，而二阶微分则细得多。

再来看孤立噪声点，注意到二阶微分对于噪声点的响应较一阶微分要强很多，这也就是图5.9中的拉普拉斯锐化图像中出现一些零星的高响应的原因，当然二阶微分的这一性质是我们所不希望的。

细线常常对应于图像中的细节，二阶微分对细线的较强响应说明了二阶微分对于细节增强的优越性。

最后，一、二阶微分对于灰度阶梯有着相同的响应，只是在二阶微分中有一个从正到负的过渡，这一性质将在第9章（边缘检测与图像分割）中用于边缘检测。

我们将这些比较的结论总结如下。

- 一阶导数通常会产生较宽的边缘。
- 二阶导数对于阶跃性边缘中心产生零交叉，而对于屋顶状边缘（细线），二阶导数取极值。
- 二阶导数对细节有较强的响应，如细线和孤立噪声点。

对于图像增强而言，基于二阶导数的算子应用更多一些，因为它对于细节响应更强，增强效果也就更明显。而在本书第9章讨论边缘检测的时候，基于一阶导数的算子则会更多地发挥作用。尽管如此，一阶算子在图像增强中依然不可或缺，它们常常同二阶算子结合在一起以达到更好的锐化增强效果。

前面最后我们提到了平滑滤波器又可以称为低通滤波器，相应的，

上面介绍的几种锐化滤波器也可以称为高通滤波器，具体得名的原因将在下一章频域滤波中说明。

5.5.5 高提升滤波及其实现

1．高提升滤波的原理

无论是基于一阶微分的Robert、Sobel模板还是基于二阶微分的拉普拉斯模板，其中各系数和均为0。这说明算子在灰度恒定区域的响应为0，即在锐化处理后的图像中，原图像的平滑区域近乎于黑色，而原图中所有的边缘、细节和灰度跳变点都作为黑背景中的高灰度部分突出显示。在基于锐化的图像增强中常常希望在增强边缘和细节的同时仍然保留原图像中的信息，而不是将平滑区域的灰度信息丢失。因此可以把原图像加上锐化后的图像得到比较理想的结果。

需要注意具有正的中心系数和具有负的中心系数的模板之间的区别。对于中心系数为负的模板（如w1, w3, w5），要达到上述的增强效果，显然应当让原图像f(i, j)减去锐化算子直接处理后的图像，即

$$g(i,\ j)\ =\ \begin{cases} f(i,j)+\mathrm{Sharpen}(f(i,j)), & if\ \text{锐化算子中心系数} > 0 \\ f(i,j)-\mathrm{Sharpen}(f(i,j)), & if\ \text{锐化算子中心系数} < 0 \end{cases} \qquad (5\text{-}13)$$

式中：Sharpen(.)表示通用的锐化算子。

这里仅以拉普拉斯锐化为例，图5.11（b）所示为形式5-13的处理效果。



（a）平滑后的婴儿照片 baby_smooth.bmp



（b）图像（a）经过式 5-13 处理

▲图5.11

图5.11（b）由于锐化后边缘和细节处的高灰度值的存在，经灰度伸缩后（归一化在[0，255]）原图灰度被压缩在一个很窄的范围内，整体上显得较暗。为了改善这种情况，对上面介绍的方法进行推广，具体的说就是在复合f(i, j)和Sharpen(f(i, j))时适当地提高f(i, j)的比重，形式化的描述如下。

$$g(i,\ j)\ =\ \begin{cases} Af(i,j)+\text{Sharpen}(f(i,j)), & if\text{锐化算子中心系数}>0 \\ Af(i,\text{j})-\text{Sharpen}(f(i,j)), & if\text{锐化算子中心系数}<0 \end{cases} \qquad (5\text{-}14)$$

形式5-14这样的滤波处理就称为高提升滤波。

一般来说权重系数A应为一个大于等于1的实数，A越大原图像所占比重越大，锐化效果越来越不明显。图5.11（b）对应于A=1的情况，图5.12（a）和（b）分别给出了当A分别为1.8和3时对于图 5.11（a）中图像的高提升滤波的效果，图中细节得到了有效的增强，对比度也有了一

定的改善。



（a）图像（a）经过高提升滤波，A=1.8



（b）图像（a）经过高提升滤波，A=3

▲图5.12 高提升滤波效果

2．高提升滤波的实现思路

高提升滤波可由以下3个步骤完成。

（1）图像锐化。

（2）原图像与锐化图像的按比例混合。

（3）混合后的灰度调整（归一化至[0, 255]）。

5.5.6 高斯-拉普拉斯变换（**Laplacian of a Gaussian, LoG**）

锐化在增强边缘和细节的同时往往也“增强”了噪声，因此如何区分开噪声和边缘是锐化中要解决的一个核心问题。

基于二阶微分的拉普拉斯算子对于细节（细线和孤立点）能产生更强的响应，并且各向同性，因此在图像增强中较一阶的梯度算子更受到我们的青睐。然而，它对于噪声点的响应也更强，我们看到对于图像

baby_noise.bmp，经拉普拉斯锐化后噪声更明显，如图5.13（b）所示。

为了在取得更好的锐化效果的同时把噪声的干扰降到最低，可以先对带有噪声的原始图像进行平滑滤波，再进行锐化增强边缘和细节。本着“强强联合”的原则，将在平滑领域工作得更好的高斯平滑算子同锐化界表现突出的拉普拉斯锐化结合起来，得到高斯-拉普拉斯算子（Marr和Hildreth提出）。

考虑高斯型函数

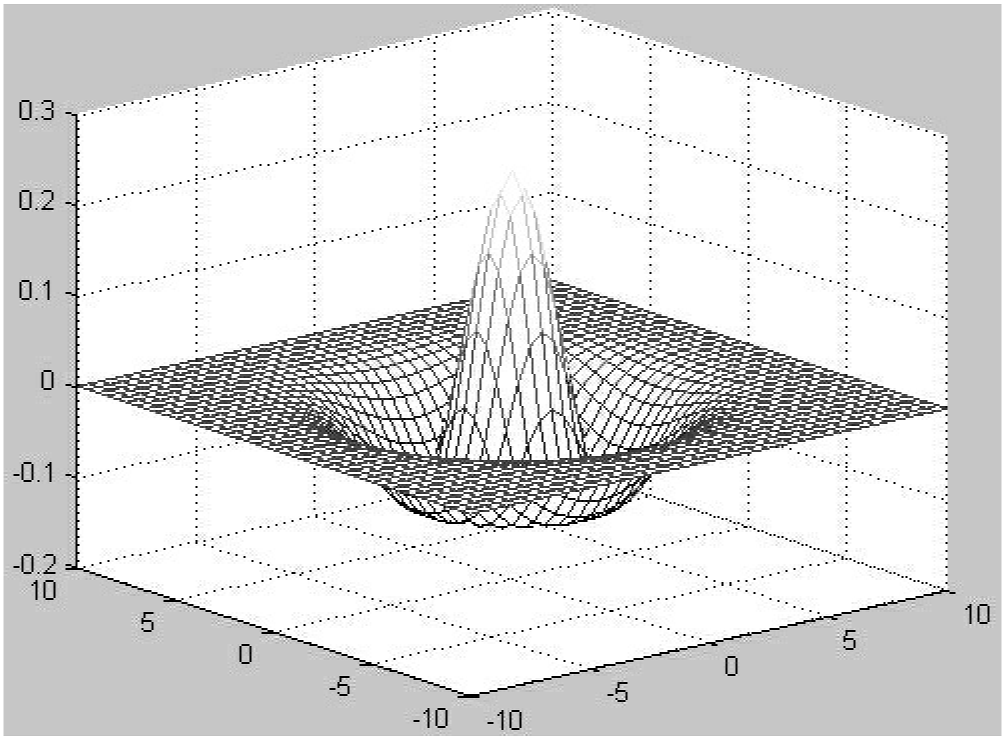
$$h(r) = -e^{-\frac{r^2}{2\sigma^2}} \tag{5-15}$$

式中：$r^2 = x^2 + y^2$；σ为标准差。

图像经该函数滤波将产生平滑效应，且平滑的程度由σ决定。进一步计算 h 的拉普拉斯算子（h关于r求二阶导数），从而得到著名的高斯-拉普拉斯算子（Laplacian of a Gaussian, LoG）。

$$\nabla^2 h(r) = -[\frac{r^2 - \sigma^2}{\sigma^4}]e^{-\frac{r^2}{2\sigma^2}} \tag{5-16}$$

图5.13展示了一个LoG函数的三维形状。如同从高斯函数得到高斯模板一样，将上式经过离散化可近似为一个5×5的拉普拉斯模板。



▲图5.13 Laplacian函数三维图形

[例 5.8] Laplacian与LoG算子的锐化效果比较。

下面给出了对于图像babyNew.bmp，分别采用Laplacian和LoG算子进行锐化的Matlab实现。

```
>> I = imread('babyNew.bmp');
>> figure, imshow(I, []); %得到图5.14（a）
>> Id = double(I); % 滤波前转化为双精度型
>> h_lap = [-1 -1 -1; -1 8 -1; -1 -1 -1] %拉普拉斯算子
h_lap =
    -1   -1   -1
    -1    8   -1
    -1   -1   -1
>> I_lap = imfilter(Id, h_lap, 'corr', 'replicate'); % Laplacian锐化
>> figure, imshow(uint8(abs(I_lap)), []); % 取绝对值并将255以上的响
```

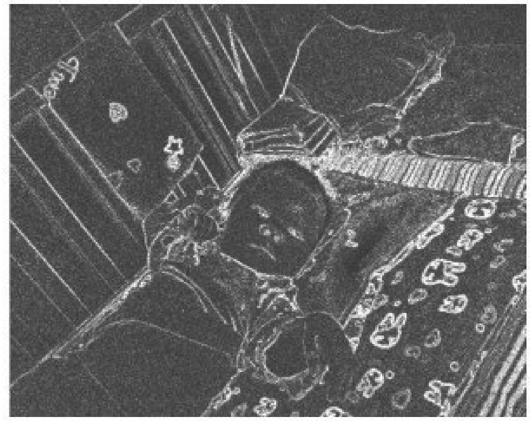
应截断, %得到图5.14（b）

```
>>
>> h_log = fspecial('log', 5, 0.5); % 大小为5，sigma=0.5的LoG算子
>> I_log = imfilter(Id, h_log, 'corr', 'replicate');
>> figure, imshow(uint8(abs(I_log)), []);%得到图5.14（c）
>>
>> h_log = fspecial('log', 5, 2); % 大小为5，sigma=2的LoG算子
>> I_log = imfilter(Id, h_log, 'corr', 'replicate');
>> figure, imshow(uint8(abs(I_log)), []);%得到图5.14（d）
```

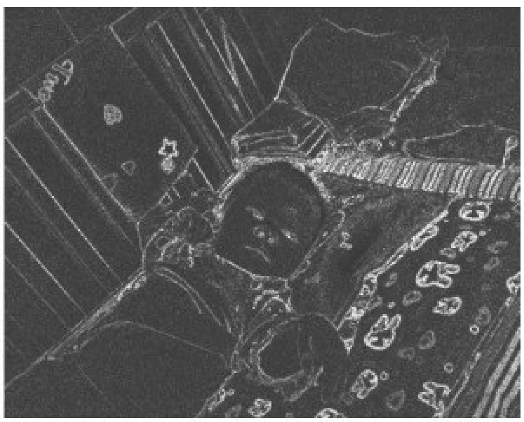
上述程序的运行结果如图 5.14 所示。图 5.14 （ c ）和图 5.14 （ d ）分别给出了对于图像babyNew.bmp，当σ=0.5和σ=2时的LoG增强效果。与图5.14（b）相比，噪声得到了有效的抑制，且σ越小细节增强效果更好，σ越大则平滑效果越好。



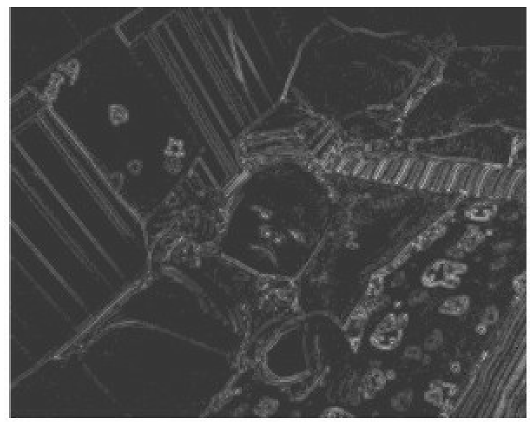
（a）原图像 babyNew.bmp



（b）Laplacian 锐化图像，噪声较明显



（c）经 LoG 处理后的图像，$\sigma = 0.5$



（d）经 LoG 处理后的图像，$\sigma = 2$

▲图5.14 Laplacian与 LoG算子滤波效果比较

关于高斯-拉普拉斯变换更为详细的内容，可参考本书第11章中关于边缘检测方法的讨论。

第6章 频率域图像增强

空间域和频率域为我们提供了不同的视角。在空间域中，函数的自变量（x，y）被视为二维空间中的一点，数字图像f（x，y）即为一个定义在二维空间中的矩形区域上的离散函数；换一个角度，如果将f（x，y）视为幅值变化的二维信号，则可以通过某些变换手段（如傅立叶变换、离散余弦变换、沃尔什变换和小波变换等）在频域下对它进行分析。

第5章详细介绍了空间域图像增强的有关知识，紧接着本章就从频域的角度去看待和分析图像增强问题，相信这一定会使您对图像增强的理解更加深刻。

本章的知识和技术热点

- 傅立叶变换的数学基础
- 快速傅立叶变换
- 频率域图像增强
- 高通滤波器和低通滤波器

本章的典型案例分析

- 美女与猫——交换两幅图像的相位谱
- 利用频域滤波消除周期噪声

6.1 频率域滤波——与空间域滤波殊途同归

在很多情况下，频率域滤波和空间域滤波可以视为对于同一个图像增强问题的殊途同归的两种解决方式。而在另外一些情况下，有些增强问题更适合在频域中完成（6.7节），有些则更适合在空域中完成。我们常常根据需要选择是工作在空间域还是频率域，并在必要时在空间域

和频率域之间相互转换。

傅立叶变换提供了一种变换到频率域的手段，由于用傅立叶变换表示的函数特征可以完全通过傅立叶反变换进行重建，不丢失任何信息，因此它可以使我们工作在频率域，而在转换回空间域时不丢失任何信息。

6.2 傅立叶变换基础知识

要理解傅立叶变换，掌握频率域滤波的思想，必要的数学知识是不能跳过的。为便于理解，我们将尽可能定性地去描述。其实傅立叶变换所必需的数学知识对于一个理工科大学二年级以上的学生来说是很有限的，高等数学中傅立叶级数的知识加上线性代数中基和向量空间的概念就足够了。下面就从一维情况下的傅立叶级数开始进行介绍。

6.2.1 傅立叶级数

法国数学家傅立叶发现任何周期函数只要满足一定条件（狄利赫里条件）都可以用正弦函数和余弦函数构成的无穷级数，即以不同频率的正弦和余弦函数的加权和来表示，后世称为傅立叶级数。

对于有限定义域的非周期函数，可以对其进行周期拓延，从而使其在整个扩展定义域上为周期函数，从而也可以展开为傅立叶级数。

1．傅立叶级数的三角形式

周期为T的函数 f (t)的三角形式傅立叶级数展开为

$$f(x)=\frac{a_0}{2}+\sum_{k=1}^{+\infty}(a_k\cos n\omega_0 x+b_k\sin n\omega_0 x) \qquad (6\text{-}1)$$

式中：

$$\omega_0 = \frac{2\pi}{T} = 2\pi u$$

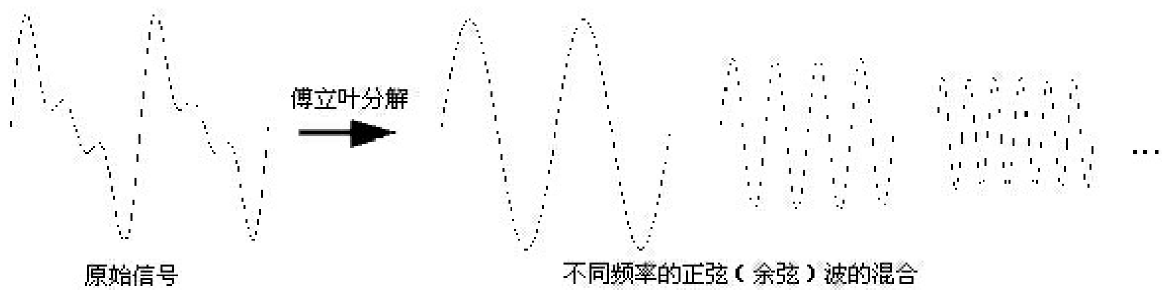
，而u=1/T，是函数f(x)的频率。

a_k和 b_k称为傅立叶系数。稍后在学习傅立叶级数的复数形式时还将介绍傅立叶系数的另一种形式。事实上，傅立叶系数正是我们在6.2.2小节傅立叶变换中所关心的对象。

于是，周期函数f(t)就与下面的傅立叶序列产生了一一对应，即

$$f(x) \Leftrightarrow \{a_0, (a_1, b_1), (a_2, b_2) \cdots\} \qquad (6\text{-}2)$$

图6.1所示形象地显示出了这种频率分解，左侧的周期函数f（x）可以由右侧函数的加权和来表示，即由不同频率的正弦和余弦函数以不同的系数组合在一起。



▲图6.1 函数f（x）的傅立叶分解。

原函数f（x）（左），其傅立叶展开为一系列不同频率的正弦、余弦函数的加权和（右）

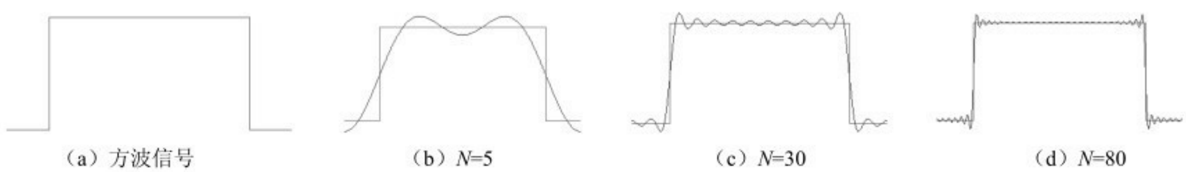
从数学上已经证明了，傅立叶级数的前N项和是原函数f(t)在给定能量下的最佳逼近。

$$\lim_{N\to\infty}\int_0^T \left| f(t) - \left[\frac{a_0}{2} + \sum_{k=1}^{N} (a_k \cos k\omega_0 t + b_k \sin k\omega_0 t) \right] \right|^2 \mathrm{d}x = 0 \qquad (6\text{-}3)$$

图6.2为我们展示了对于一个方波信号函数采用不同的N值的逼近情况。随着N的增大，逼近效果越来越好。但同时也注意到，在f (x)的不可导点上，如果只取式（6-1）右边的无穷级数中的有限项之和作为$\hat{f}(x)$，那么$\hat{f}(x)$在这些点上会有起伏，对于图6.2（a）的方波信号尤为明显，这就是著名的吉布斯现象。

2．傅立叶级数的复指数形式

除上面介绍的三角形式外，傅立叶级数还有其他两种常用的表现形式，即余弦形式和复指数形式。借助欧拉公式，上述3种形式可以很方便地进行等价转化，本质上它们都是一样的。



▲图6.2 采用不同的N值时傅立叶级数展开的逼近效果

复指数傅立叶级数即我们经常说的傅立叶级数的复数形式，因其具有简洁的形式（只需一个统一的表达式计算傅立叶系数），在进行信号和系统分析时通常更易于使用；而余弦傅立叶级数可使周期信号的幅度谱和相位谱意义更加直观，函数的余弦傅立叶级数展开可以解释为f(x)可以由不同频率和相位的余弦波以不同系数组合在一起来表示，而在三角形式中相位是隐藏在系数A_n和b_n中的。下面主要介绍复指数傅立叶级数，在后面的傅立叶变换中要用到的正是这种形式。关于余弦傅立叶级数的有关知识，感兴趣的读者请参考附录III。

傅立叶级数的复指数形式为

$$f(x)=\sum_{n=-\infty}^{\infty} c_n \mathrm{e}^{i2n\pi ux} \qquad （6\text{-}4）$$

式中：

$$c_n = \frac{1}{T}\int_{-T/2}^{T/2} f(x)\mathrm{e}^{-i2n\pi ux}\mathrm{d}x \qquad (n=0,\pm1,\pm2,\ldots) \tag{6-5}$$

由式（6-4）和（6-5）可见，复指数傅立叶级数形式比较简洁，级数和系数都可以采用一个统一的公式计算。有关如何由式（6-1）推导出傅立叶级数复指数形式（6-4）的过程，由于这里我们感兴趣的并非傅立叶级数本身，就不在正文中给出了，详细的内容可参考附录Ⅱ，只要您相信不同的展开形式之间本质上是等价的，并对复指数形式的傅立叶级数展开建立了一个基本的形式上的认识就足以继续阅读和理解后面的内容了。

6.2.2 傅立叶变换

1．一维连续傅立叶变换

对于定义域为整个时间轴（−∞< t <∞）的非周期函数 f（t），此时已无法通过周期拓延将其扩展为周期函数，这种情况下就要用到傅立叶变换。

$$F(u) = \int_{-\infty}^{\infty} f(x)\mathrm{e}^{-i2\pi ux}\mathrm{d}x \tag{6-6}$$

由F(u)我们还可以通过傅立叶反变换获得f（t）。

$$f(x) = \int_{-\infty}^{\infty} F(u)\mathrm{e}^{i2\pi ux}\mathrm{d}u \tag{6-7}$$

式（6-6）和式（6-7）即为我们通常所说的傅立叶变换对，6.1 节中提到的函数可以从它的反变换进行重建正是基于上面的傅立叶变换对。

由于傅立叶变换与傅立叶级数涉及两类不同的函数，在很多数字图

像处理的书中通常对它们分别进行处理，并没有阐明它们之间存在的密切联系，这给很多初学者带来了困扰，实际上我们不妨认为周期函数的周期可以趋向无穷大，这样可以将傅立叶变换看成是傅立叶级数的推广。

仔细地观察式（6-6）和式（6-7），对比复指数形式的傅立叶级数展开公式式（6-4），注意到在这里傅立叶变换的结果F(u)实际上相当于傅立叶级数展开中的傅立叶系数，而反变换公式式（6-7）则体现出不同频率复指数函数的加权和的形式，相当于复指数形式的傅立叶级数展开公式，只不过这里的频率u变为了连续的，所以加权和采用了积分的形式。这是因为随着作为式（6-5）的积分上下限的T向整个实数定义域扩展，即$T\to\infty$,频率u则趋近于du（因为u=1/T），导致原来离散变化的u的连续化。

2．一维离散傅立叶变换

一维函数 f(x)（其中 x=0, 1, 2 ,… ,M-1）的傅立叶变换的离散形式为

$$F(u)=\sum_{x=0}^{M-1}f(x)e^{-i2\pi ux/M}, u=0,1,2,\ldots,M-1 \quad (6\text{-}8)$$

相应的反变换为

$$f(x)=\frac{1}{M}\sum_{u=0}^{M-1}F(u)e^{i2\pi ux/M}, x=0,1,2,\ldots,M-1 \quad (6\text{-}9)$$

由于一维情况下很多性质更为直观，我们更青睐于分析一维离散傅立叶变换，而由此得出的这些结论都可顺利推广至二维。一些有用的性质如下。

● 仔细观察式（6-8）和式（6-9），注意到在频域下变换F(u)也是

离散的，且其定义域仍为0～M−1，这是因为F(u)的周期性，即

$$F(u+M)=F(u) \tag{6-10}$$

● 考虑式（6-9）中的系数1/M，在这里该系数被放在反变换之前，实际上它也可以位于式（6-8）的正变换公式中。更一般的情况是只要能够保证正变换与反变换之前的系数乘积为1/M即可。例如，两个公式的系数可以均为$1/\sqrt{M}$。

● 为了求得每一个F(u)（u=0, 1, 2,…,M-1），都需要全部M个点的f(x)参与加权求和计算。对于M个u，则总共需要大约M^2次计算。对于比较大M（在二维情况下对应着比较大的图像），计算代价还是相当可观的，我们会在下一节快速傅立叶变换中来研究如何提高计算效率的问题。

3．二维连续傅立叶变换

有了之前的基础，下面我们将傅立叶变换及其反变换推广至二维。对于二维连续函数，傅立叶变换为

$$F(u,v)=\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}f(x,y)\mathrm{e}^{-\mathrm{j}2\pi(ux+vy)}\mathrm{d}x\mathrm{d}y \tag{6-11}$$

类似的，其反变换为

$$f(x,y)=\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}F(u,v)e^{\mathrm{j}2\pi(ux+vy)}\mathrm{d}u\mathrm{d}v \tag{6-12}$$

4．二维离散傅立叶变换

在数字图像处理中，我们关心的自然是二维离散函数的傅立叶变换，下面直接给出二维离散傅立叶变换（Discrete Fourier Transform, DFT）公式。

$$F(u,v)=\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}f(x,y)\mathrm{e}^{-i2\pi(ux/M+vy/N)} \tag{6-13}$$

$$f(x,y)=\frac{1}{MN}\sum_{u=0}^{M-1}\sum_{v=0}^{N-1}F(u,v)\mathrm{e}^{i2\pi(ux/M+vy/N)} \tag{6-14}$$

相对于空间域（图像域）的变量x、y，这里的u、v是变换域或者说是频率域变量。同一维中的情况相同，由于频谱的周期性式（6-13）只需对u值（u=0, 1, 2, …,M−1）及 v值（v=0, 1, 2, …N−1）进行计算。同样，系数1/MN的位置并不重要，有时也放在正变换之前，有时则在正变换和反变换前均乘以系数$1/\sqrt{MN}$。

根据式（6-13），频域原点位置的傅立叶变换为

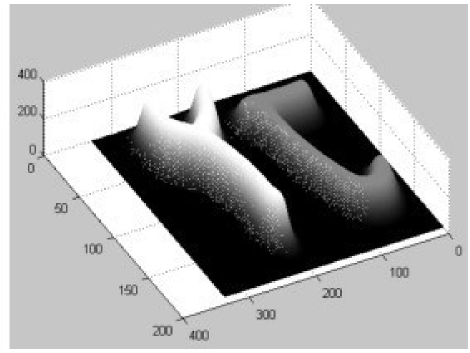
$$F(0,0)=\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}f(x,y) \tag{6-15}$$

显然，这是f(x, y)各个像素的灰度之和。而如果将系数1/MN放在正变换之前，则F(0, 0)对应于原图像f(x, y)的平均灰度。F(0, 0)有时被称作频谱的直流分量（DC）。

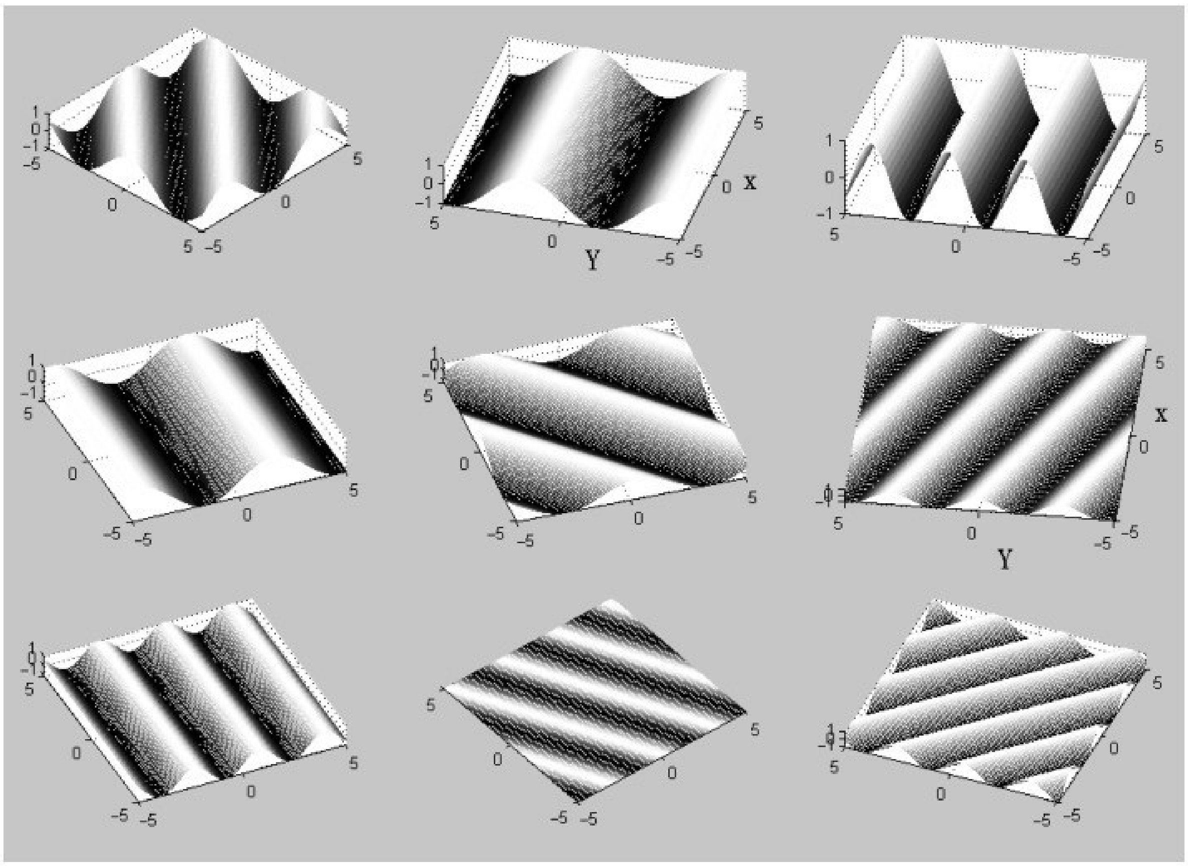
我们之前曾指出了一维函数可以表示为正弦（余弦）函数的加权和形式；类似的，二维函数f(x, y)可以分解为不同频率的二维正弦（余弦）平面波的按比例叠加。图6.3（a）中给出了一幅简单的图像，可将它视为以其灰度值作为幅值的二维函数，如图 6.3（b）所示，根据式（6-13），它可以分解为如图6.3（c）所示的不同频率和方向的正弦（余弦）平面波的按比例叠加（只给出了一部分）。比如图6.3（c）中第一行中间的平面波为sin(Y)，而第二行右面的平面波则为sin(X+2Y)，而第三行最后的一个为sin(2X+2Y)。



（a）原图像



（b）图像（a）对应的二维函数



（c）二维傅立叶变换的基函数——不同频率和方向的正弦平面波（不同视角）

▲图6.3 二维函数 f(x, y)的傅立叶分解

6.2.3 幅度谱、相位谱和功率谱

下面，我们再来定义傅立叶变换的幅度谱、相位谱以及功率谱。

● 幅度谱

$$|F(u,v)|=[\mathrm{Re}(u,v)^2+\mathrm{Im}(u,v)^2]^{1/2} \qquad (6\text{-}16)$$

显然，幅度谱关于原点具有对称性，即|F (−u,−v)|=|F (u, v)|。

● 相位谱

$$\varphi(u,v)=\arg\tan\frac{\mathrm{Im}(u,v)}{\mathrm{Re}(u,v)} \qquad (6\text{-}17)$$

通过幅度谱和相位谱，我们可以还原F (u, v)。

$$F(u,v)=|F(u,v)|\,\mathrm{e}^{\mathrm{j}\varphi(u,v)} \qquad (6\text{-}18)$$

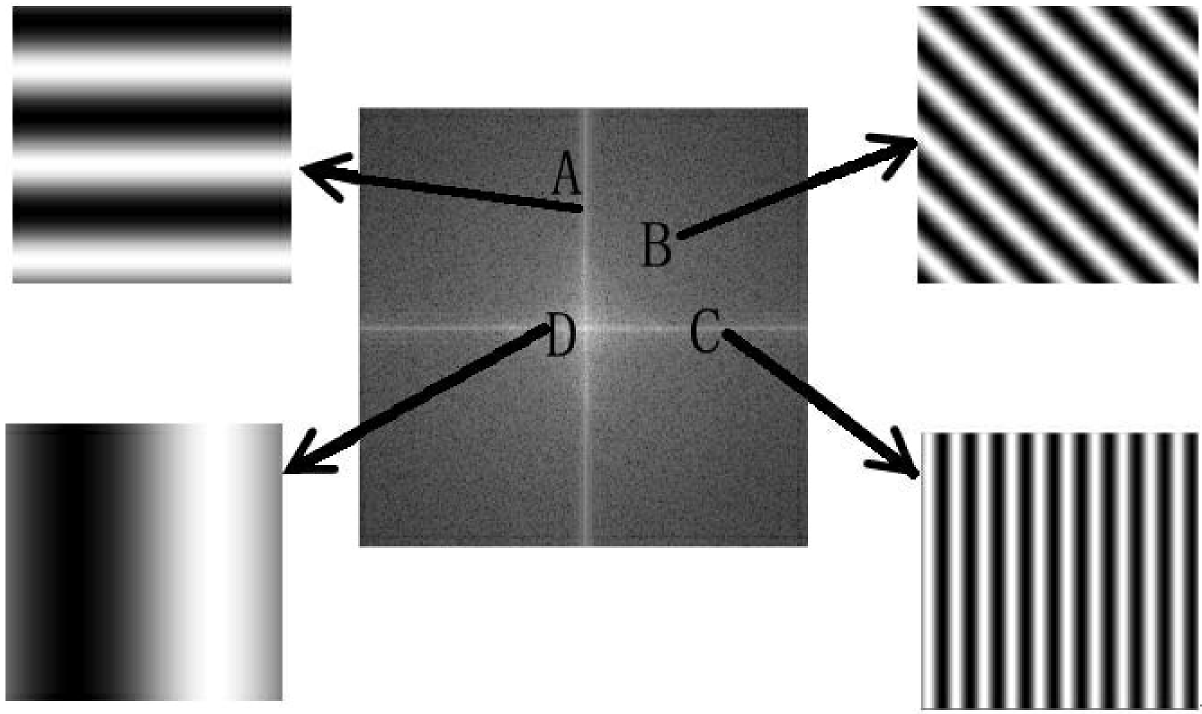
● 功率谱（谱密度）

$$P(u,v)=|F(u,v)|^2=\mathrm{Re}(u,v)^2+\mathrm{Im}(u,v)^2 \qquad (6\text{-}19)$$

式中：Re(u, v)和Im(u, v)分别为F(u,v)的实部和虚部。

幅度谱又叫频率谱，是图像增强中关心的主要对象，频域下每一点（u,v）的幅度|F(u, v)|可用来表示该频率的正弦（余弦）平面波在叠加中所占的比例，如图6.4所示。幅度谱直接反映频率信息，是频域滤波中的一个主要依据。

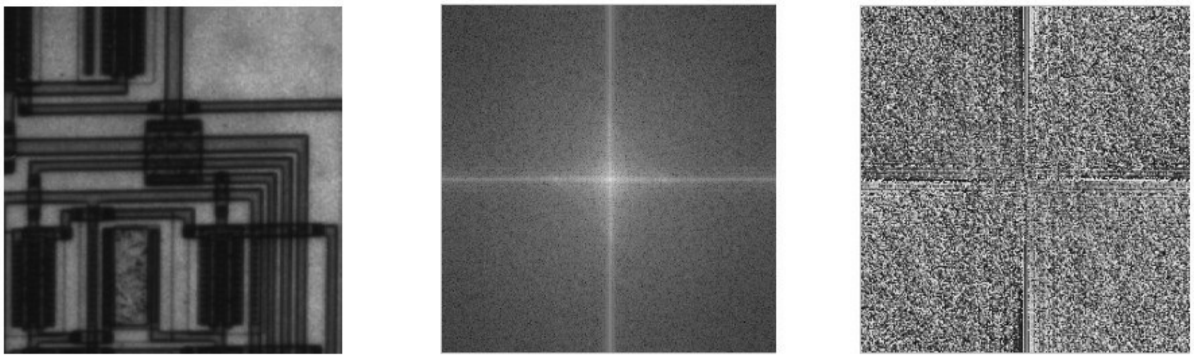
图6.4所示幅度谱中的A、B、C、D四点的幅值分别为四周的 4 个正弦平面波在的加权求和中的权值（混合比例）。注意这4个正弦平面波的方向和频率。



▲图6.4 幅度谱的意义

相位谱表面上看并不那么直观，但它隐含着实部与虚部之间的某种比例关系，因此与图像结构息息相关。

由于对于和空域等大的频域空间下的每一点（u, v），均可计算一个对应的|F(u, v)|和φ(u, v)，因此可以像显示一幅图像那样显示幅度谱和相位谱。图6.5（b）、（c）分别给出了图6.5（a）中图像的幅度谱和相位谱，获得它们的方法请参考6.3节中傅立叶变换实现的相关内容，关于幅度谱和相位谱的一个非常有趣的例子请参考例6.2。



▲图6.5 circuit.tif幅度谱和相位谱。幅度谱和相位谱都将（0,0）点移到了中心

6.2.4 傅立叶变换的实质—基的转换

无论是傅立叶变换、离散余弦变换还是小波变换，其本质都是基的变换。下面首先让我们一起回顾一下线性代数中基和向量空间的相关知识。

1．基和向量空间

在三维欧氏向量空间中，某向量$\vec{v}$可以由3个复数$\{v_1,v_2,v_3\}$来定义，常常记作$\vec{v}=(v_1,v_2,v_3)$，这3个复数与3个正交单位向量$\{\vec{e}_1,\vec{e}_2,\vec{e}_3\}$相联系。实际上，有序集$\{v_1,v_2,v_3\}$表示向量$\vec{v}$的3个标量分量，也就是系数；而3个正交单位向量$\{\vec{e}_1,\vec{e}_2,\vec{e}_3\}$即为该三维欧氏空间的基向量。我们称该空间为这3个基向量所张成的空间，任何该空间中的向量$\vec{v}$均可由这3个基向量的线性组合（加权和）表示为

$$\vec{v}=v_1\vec{e}_1+v_2\vec{e}_2+v_3\vec{e}_3=\sum_{i=1}^{3}v_i\vec{e}_i \tag{6-20}$$

也可以用矩阵的形式来表示该向量。

$$\vec{v}=\begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} \tag{6-21}$$

在上面的叙述中涉及了向量的正交，这是向量代数中一个非常重要的概念。为了说明正交的概念，让我们首先回顾一下向量点积（数量积），两个向量的点积定义为

$$\vec{u}\bullet\vec{v}=|\vec{u}||\vec{v}|\cos\theta=u_1v_1+u_2v_2+u_3v_3=[u_1\ u_2\ u_3]\begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix}=\begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix}^{\mathrm{T}}\begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix}=\vec{u}^{\mathrm{T}}\vec{v} \tag{6-22}$$

式中：$|\vec{v}|=\sqrt{{v_1}^2+{v_2}^2+{v_3}^2}$，表示向量$\vec{v}$的模；θ为向量$\vec{u}$和$\vec{v}$之间的夹角；上标T表示转置。

此时，如果$\vec{u}\bullet\vec{v}$=0，则称这两个向量$\vec{u}$和$\vec{v}$互相正交。由式（6-22）可知，两非零向量正交则cosθ=0，说明其夹角为90°（垂直）。

接下来，定义一个向量在另一个向量方向上的投影或分量为

$$\vec{u}\text{在}\vec{v}\text{方向上的投影（分量）}=\vec{u}\bullet\frac{\vec{v}}{|\vec{v}|}=\vec{u}\bullet\vec{e}_v \tag{6-23}$$

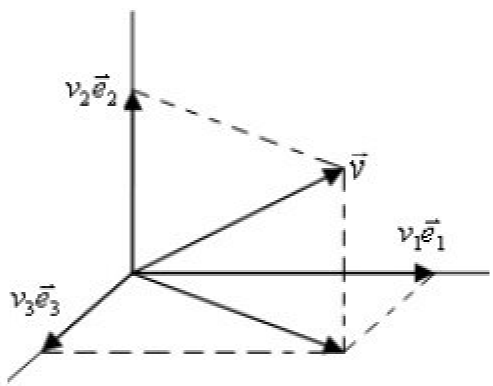
式中：$\vec{e}_v$为向量$\vec{v}$单位化后的单位向量，模为1，方向与$\vec{v}$相同。式（6-23）说明如果需要得到某向量在给定方向上的分量，只需计算该向量与给定方向单位向量的点积。

图6.6能够帮助我们理解上述内容，图6.6（a）中为一个三维空间中的向量$\vec{v}$以及3个单位正交基向量$\vec{e}_1,\vec{e}_2,\vec{e}_3$；图6.6（b）中给出了向量$\vec{v}$在$\vec{e}_2$方向的投影$v_2$；在图6.6（c）中，根据矢量加法的平行四边形法则，向量$\vec{v}$被分解为3个正交基向量$\vec{e}_1,\vec{e}_2,\vec{e}_3$线性组合，显然可以表示为$\vec{v}$=$(v_1,v_2,v_3)$的形式。



（a）向量 $\vec{v}$ 和三个正交基向量 $\vec{e}_1,\vec{e}_2,\vec{e}_3$　　（b）向量 $\vec{v}$ 在 $\vec{e}_2$ 方向的投影 v_2

▲图6.6 三维欧几里得空间中向量的投影和正交分解



（c）将向量 $\vec{v}$ 分解为三个正交基向量 $\vec{e}_1,\vec{e}_2,\vec{e}_3$ 线性组合的形式：$\vec{v}=\sum_{i=1}^{3}v_i\vec{e}_i$

▲图6.6 三维欧几里得空间中向量的投影和正交分解（续）

将三维向量空间中基与投影的概念推广至N维向量空间。任何一个该空间中的N×1向量均可由N个基向量$\vec{e}_1$,$\vec{e}_2$,...,$\vec{e}_N$的线性组合来表示，记作

$$\vec{v}=\sum_{i=1}^{N}v_i\vec{e}_i \tag{6-24}$$

式中：分量v_i为向量$\vec{v}$在$\vec{e}_i$方向的投影。

$$v_i=\vec{v}\bullet\vec{e}_i \tag{6-25}$$

式（6-24）称为对$\vec{v}$的重构，式（6-25）称为对$\vec{v}$的分解。

而N个单位基向量之间满足两两正交关系，即

$$\vec{e}_i \bullet \vec{e}_j = \delta_{i,j} = \begin{cases} 1, i = j \\ 0, i \neq j \end{cases} \quad \forall i, j \in Z \tag{6-26}$$

2．基函数和函数空间

尽管上面的向量分解与重构的问题比较基础，但它与傅立叶变换与反变换之间的关系却十分紧密。事实上，它们在形式上有着惊人的相似，唯一不同的是这里的向量空间变成了函数空间，向量$\vec{v}$变成了函数f(x)，而基向量$\vec{e}_1, \vec{e}_2, \ldots, \vec{e}_n$也相应地变成了基函数。对比式（6-24）～（6-25）和式（6-8）～（6-9）的形式不难看出，式（6-25）的分解过程即相当于傅立叶变换，而式（6-26）的重构过程则恰恰相当于傅立叶反变换。也就是说，相应函数空间中的任意函数均可以由该函数空间中的一组基函数的加权和来表示。观察式（6-8）容易发现，这里的基函数的形式为$e^{-i\,2\pi ux}$，我们用下面的等式来表示函数的正交性。

$$\frac{1}{T}\int_{-T/2}^{T/2} e^{i2\pi kx} e^{-i2\pi lx} = \delta_{k,l} \qquad \forall k, l \in Z \tag{6-27}$$

至此，读者应该已经理解了傅立叶变换的实质——基的转换。对于给定函数f(x)，关键是选择合适的基，使得f(x)在这组基下表现出我们需要的特性。当某一组基不满足要求时，就需要通过变换将函数转换到另一组基下表示，方可得到我们需要的函数表示。常用的变换有傅立叶变换（以正弦和余弦函数为基函数）、小波变换（以各种小波函数为基函数）、离散余弦变换以及 Walsh 变换等。实际上，我们在第12章中将指出，特征降维中常用的主成份分析法（K-L变换）本质上也是一种基的转换。

6.3 快速傅立叶变换及实现

6.2节介绍了离散傅立叶变换（DFT）的原理，但并没有涉及其实现问题，这主要是因为DFT的直接实现效率较低。在工程实践中，我们迫切地需要一种能够快速计算离散傅立叶变换的高效算法，快速傅立叶变换（FFT）便应运而生。本节将给出快速傅立叶变换算法的原理及其实现细节。

6.3.1 FFT变换的必要性

之所以提出快速傅立叶变换（FFT）方法，是因为在计算离散域上的傅立叶变换时，对于N点序列，它的DFT变换与反变换对定义为

$$\begin{cases} F(u)=\sum_{x=0}^{N-1} f(x)W_N^{ux},\ u=0,1,\dots,N-1, W_N=\mathrm{e}^{-\mathrm{j}\frac{2\pi}{N}} \\ f(x)=\frac{1}{N}\sum_{u=0}^{N-1} F(u)W_N^{-ux},\ x=0,1,\dots,N-1 \end{cases} \tag{6-28}$$

于是不难发现，计算每个u值对应的F(u)需要N次复数乘法和N-1次复数加法。因此，为了计算长度为N的序列的快速傅里叶变换，共需要执行n^2 次复数乘法和N(N-1)次复数加法。而实现1次复数相加至少需要执行2次实数加法，执行1次复数相乘则可能需要至多4次实数乘法和2次实数加法。如果使用这样的算法直接处理图像数据，则运算量会大得惊人，更无法实现实时处理。

然而，离散傅立叶变换的计算实质并没有那么复杂。在离散傅里叶变换的运算中有大量重复运算。上面的变量W_N是一个复变量，但是可以看出它具有一定的周期性，实际上它只有N个独立的值。而这N个值也不是完全相互独立的，它们又具有一定的对称关系。关于变量W_N的

周期性和对称性，我们可以做如下总结。

$$W_N{}^0 = 1, W_N{}^{\frac{N}{2}} = -1 \qquad （6-29）$$

$$W_N{}^{N+r} = W_N{}^r, W_N{}^{\frac{N}{2}+r} = -W_N{}^r \qquad （6-30）$$

式（6-29）是W矩阵中元素的某些特殊值，而式（6-30）则说明了W矩阵元素的周期性和对称性。利用W的周期性，DFT运算中的某些项就可以合并；而利用W的对称性，则可以仅计算半个W序列。而根据这两点，我们就可以将一个长度为N的序列分解成两个长度为N/2的序列并分别计算DFT，这样就可以节省大量的运算量。我们将在讲述常见的FFT算法后分析节省的运算量。

这正是快速傅立叶变换（Fast Fourier Transform，FFT）的基本思路——通过将较长的序列转换成相对短得多的序列来大大减少运算量。

6.3.2 常见的FFT算法

目前流行的大多数成熟的 FFT 算法的基本思路大致可以分为两大类，一类是按时间抽取的快速傅立叶算法（Decimation In Time, DIT-FFT），另一类是按频率抽取的快速傅立叶算法（Decimation In Freqency, DIF-FFT）。这两种算法思路的基本区别如下。

按时间抽取的FFT算法是基于将输入序列f(x)分解（抽取）成较短的序列，然后从这些序列的DFT 中求得输入序列的 F(u)的方法。由于抽取后的较短序列仍然可分，所以最终仅仅需要计算一个很短的序列的DFT。在这种算法中，我们主要关注的是当序列的长度是 2 的整数次幂时，如何能够高效地进行抽取和运算的方法。

而按频率抽取的FFT算法是基于将输出序列F(u)分解（抽取）成较短的序列，并且从f(x)计算这些分解后的序列的 DFT。同样，这些序列

可以继续分解下去，继续得到更短的序列，从而可以更简便地进行运算。这种算法同样是主要针对2的整数次幂长度的序列的。

从本章前面对DFT的介绍和本节开头的分析可知，随着序列长度的减小，FFT运算的复杂度将以指数规律降低。

本节主要讨论序列长度是2的整数次幂时的DFT运算，这称为基−2FFT。除了基−2FFT，还有基4−FFT和基−8FFT，甚至还有基−6FFT。那些算法的效率比基−2FFT更高，但应用的范围更狭窄。事实上，很多商业化的信号分析库都是使用混合基FFT的。那样的程序代码更加复杂，但效率却高得多，而且应用范围更广。本书从学习和研究的角度，仅介绍最常见的按时间抽取的基−2FFT算法。

6.3.3 按时间抽取的基−2 FFT算法

对于基−2的FFT，可以设序列长度为$N=2^L$。由于N是偶数，我们可将这个序列按照项数的奇偶分成2组。分组的规律如下式所示。

$$\begin{cases} f(2x)=f_{偶}(x) \\ f(2x+1)=f_{奇}(x) \end{cases}, x=0,1,2,\ldots,\frac{N}{2}-1 \quad （6-31）$$

则f(x)的傅立叶变换F(u)可以表示为f(x)的奇数项和偶数项分别组成的序列的如下变换形式。

$$F(u)=\sum_{x=0}^{N-1} f(x)W_N^{ux}\sum_{x偶} f(x)W_N^{ux}+\sum_{x奇} f(x)W_N^{ux} \quad （6-32）$$

$$=\sum_{r=0}^{\frac{N}{2}-1} f(2r){W_N}^{2ru}+\sum_{r=0}^{\frac{N}{2}-1} f(2r+1){W_N}^{(2r+1)u}, r=0,1,2,\ldots,\frac{N}{2}-1 \quad （6-33）$$

因为 ${W_N}^{2nx}={W_{\frac{N}{2}}}^{nx}$ ，所以上式可以继续化简为

$$F(u)=\sum_{r=0}^{\frac{N}{2}-1} f(2r)W_{\frac{N}{2}}{}^{ru}+W_N{}^u\sum_{r=0}^{\frac{N}{2}-1} f(2r+1)W_{\frac{N}{2}}{}^{ru} \qquad (6\text{-}34)$$

容易发现，上式的第一项为f(2r)的N/2点DFT，而第二项的求和部分为f(2r+1)的N/2点DFT （序列f(2r)和序列f(2r+1)的周期均为N/2）。也即

$$F(u)=F_{偶}(u)+W_N{}^uF_{奇}(u),u=0,1,2,...,\frac{N}{2}-1 \qquad (6\text{-}35)$$

这里，我们用$f_{偶}(u)$和$f_{奇}(u)$分别表示f(2r)和f(2r+1)的N/2点DFT。

而且，根据DFT序列的周期性特点，还可得到如下式子成立。

$$F_{偶}(u)=F_{偶}(u+\frac{N}{2}),F_{奇}(u)=F_{奇数}(u+\frac{N}{2}) \qquad (6\text{-}36)$$

并且，由于 $W_N{}^{\frac{N}{2}}=-1$ ，我们还可以得出

$$W_N{}^{u+\frac{N}{2}}=W_N{}^uW_N{}^{\frac{N}{2}}=-W_N{}^u \qquad (6\text{-}37)$$

因此

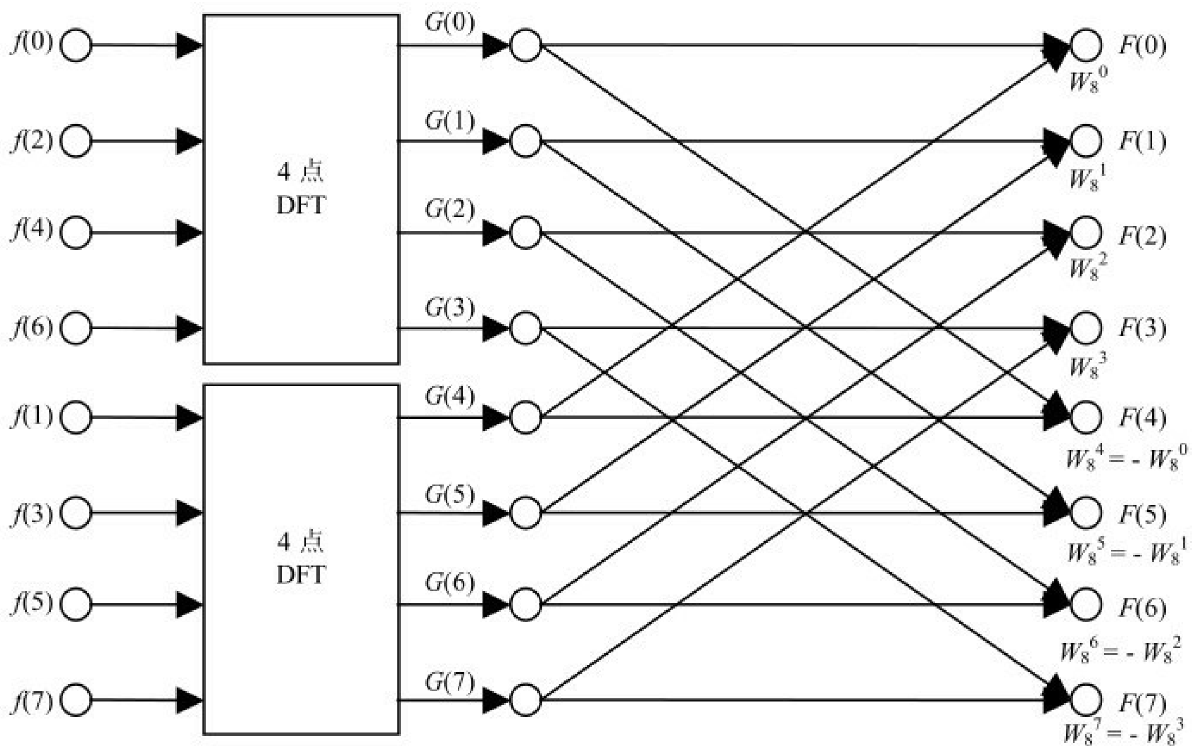
$$W_N{}^{u+\frac{N}{2}}F_{奇}(u+\frac{N}{2})=-W_N^uF_{奇}(u) \qquad (6\text{-}38)$$

将式（6-36）和式（6-37）代入式（6-38），并根据式（6-35），得

$$
\begin{cases}
F_n(u)=F_{偶}(u)+W_N{}^{u}F_{奇}(u), u=0,1,2,\ldots,\dfrac{N}{2}-1 \\
F_n(u+\dfrac{N}{2})=F_{偶}(u)+W_N{}^{u+\frac{N}{2}}F_{奇}(u+\dfrac{N}{2})=F_{偶}(u)-W_N{}^{u}F_{奇}(u), u=0,1,2,\ldots,\dfrac{N}{2}-1
\end{cases}
\quad (6\text{-}39)
$$

这是一个递推公式，它就是 FFT 蝶形运算的理论依据。该公式表明，一个偶数长度序列的傅立叶变换可以通过它的奇数项和偶数项的傅立叶变换得到，从而可以将输入序列分成两部分分别计算，并按公式相加/相减。而在这个运算过程中，实际上只需要计算 $W_N{}^{u}$，u=0,1,2,3,… N/2−1。

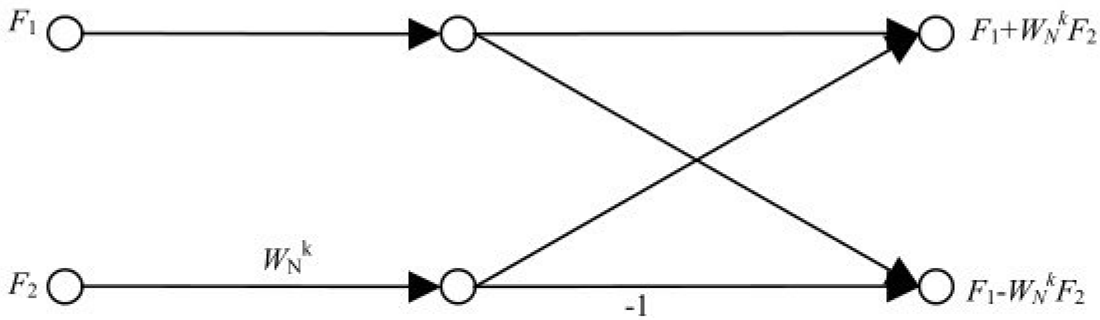
因此，一个8点按时间抽取的FFT算法的第一步骤如图6.7所示。



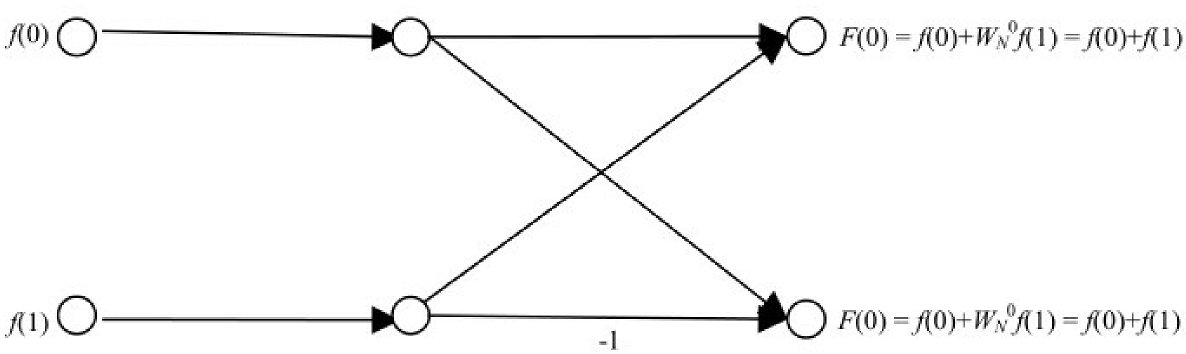
▲图6.7 8点 FFT变换简图

图6.7所示是根据式（6-39）绘制的，这一算法也可以用图6.8抽象地表示出来。

由于我们讨论的是基-2 的 FFT 算法，N/2 一般应是偶数，因此得到的序列还可以继续分解，分解过程可以一直持续到每个序列只需要2点的DFT。这样只需要如下的运算即可计算这一DFT值，这一运算是FFT的基本运算，称为蝶形运算。



▲图6.8 蝶形算法抽象示意图

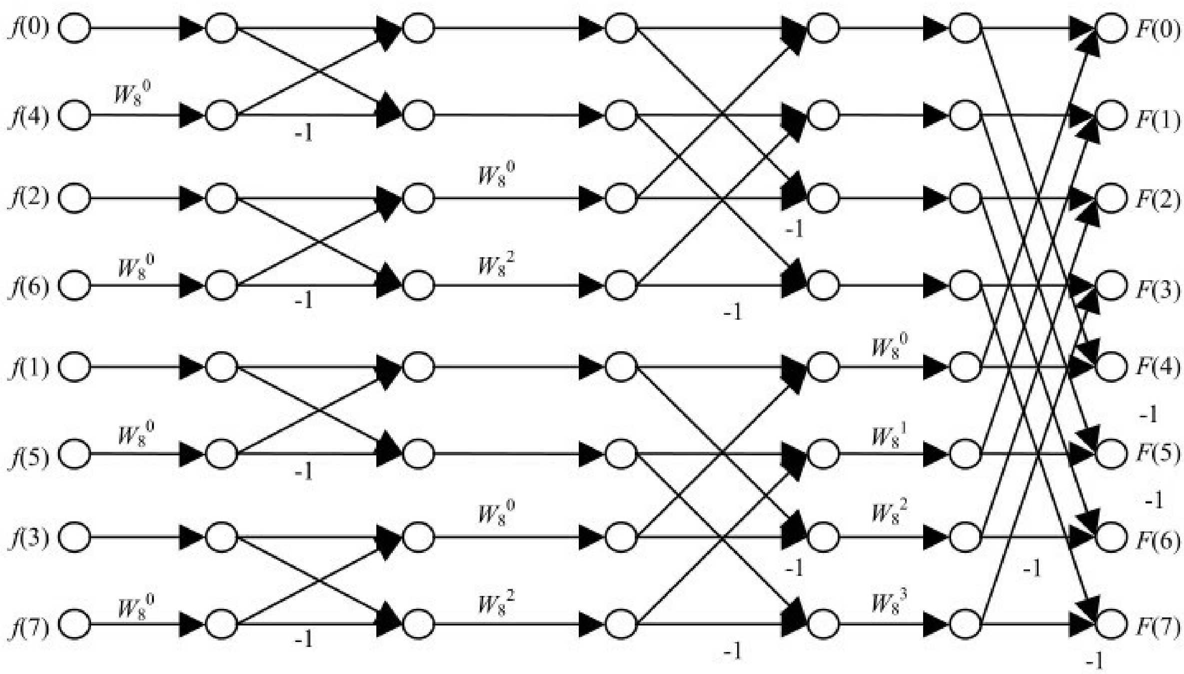


▲图6.9 蝶形算法基础单元

这一基础单元是对初始输入序列进行傅立叶变换操作的第一步，即2点时的FFT。把这个基本的DFT运算和上面的抽象化蝶形运算比较，可以发现它们的基本结构是完全一致的。在蝶形算法中，我们可以只计算一次$W_N^kF_2$，而后分别与f_1相加和相减，从而每一次蝶形算法只需1次复数乘法和2次复数加法（从复杂度分析的角度，相减当然也可看作是一次加法）。并且，注意到W_N^K=1，因此可以进一步简化计算。尤其第一

级蝶形运算更是可以完全简化为单纯的复数加减法。

一个8点FFT的完整计算过程如图6.10所示，请思考这个过程与DFT过程的区别，以及这个过程所需的算法复杂度和存储空间问题。稍后我们将讨论这个问题。



▲图6.10 8点 FFT算法

用基-2的时间抽取FFT算法比直接计算DFT的效率高得多。在计算长度为$N=2^L$ 序列的FFT时，在不对复数乘法进行额外优化的情况下，所需运算量分析如下。

对于每一个蝶形运算，我们需要进行1次复数乘法和2次复数加法。而FFT运算的每一级都含有$N/2 = 2^{L-1}$ 个蝶形运算单元。因此，完成L级FFT运算共需要的复数乘法次数m_{cm}和复数加法m_{ca}数目分别为

$$M_{cm} = L\frac{N}{2} = \frac{N}{2}\log_2 N \qquad (6\text{-}40)$$

$$M_{ca} = LN = N\log_2 N \qquad (6\text{-}41)$$

而本节开头提到，实现同样长度序列的DFT运算则需要n^2 次复数乘法和N(N-1)次复数加法，这远远多出FFT算法的所需。近似比较FFT和DFT运算的算法复杂度可知

$$C(N) = \frac{N^2}{N\log_2 N} = \frac{N}{\log_2 N} \qquad (6\text{-}42)$$

或

$$C(L) = \frac{2^L}{L} \qquad (6\text{-}43)$$

因此，在N或L取值增大时，FFT运算的优势更加明显。例如，当$N=2^{10}$时，C(N)=102.4，即FFT算法的速度是DFT的102.4倍。

此外，从占用的存储空间看，按时间抽取的FFT算法也远比DFT算法节约。一对复数进行完蝶形运算后，就没有必要再次保留输入的复数对。因此，输出对可以和输入对放在相同的存储单元中。所以，只需要和输入序列大小相等的存储单元即可。也就是一种“原位运算”。

但是，经过观察上面的8点FFT运算全过程，我们发现，如果要使用这种“原位”运算，输入序列就必须按照倒序存储。由于f(x)是逐次抽取的，所以必须对原输入码列倒转位序，得到的次序相当于是原序列编号的二进制码位倒置。也即将原序列编号按照二进制表示，并且将二进制的所有位次序颠倒，就得到了在实际的输入序列中应该使用的排序位置。下面同样以8点FFT为例说明码位倒置的方法，如表6.1所示。

表6.1 8点FFT的码位倒置对应表

n	对应的二进制数	码位倒置后的二进制数	对应在实际输入序列中的位置
0	000	000	0
1	001	100	4
2	010	010	2
3	011	110	6
4	100	001	1
5	101	101	5
6	110	011	3
7	111	111	7

按照表6.1中的顺序排列输入数据，就可以方便地进行原位运算，以节约内存空间。

6.3.4 离散反傅立叶变换的快速算法

离散反傅立叶变换的形式与离散傅立叶变换很相似,首先比较它们的公式形式。

离散反傅立叶变换IDFT的公式为

$$f(x)=\mathrm{IDFT}(F(u))=\frac{1}{N}\sum_{u=0}^{N-1}F(u)W_N^{-ux} \qquad (6\text{-}44)$$

离散傅立叶变换DFT的公式为

$$F(u)=\mathrm{DFT}(f(x))=\sum_{x=0}^{N-1}f(x)W_N^{ux} \qquad (6\text{-}45)$$

观察式（6-44）和式（6-45）发现，只要把DFT算子中的W_N^{ux}换成W_N^{-ux}，并在前面乘以1/N，即可得到IDFT的算子。于是考虑使用复数共轭方式建立两者之间的联系，推导离散反傅立叶变换的公式如下。

$$f(x)=\frac{1}{N}[\sum_{u=0}^{N-1}F^*(u)W_N^{nu}]^*=\frac{1}{N}\{\mathrm{DFT}[F^*(u)]\}^* \quad (6\text{-}46)$$

因此，我们只需先将F(u)取共轭，就可以直接使用FFT算法计算IFFT了。

6.3.5 N维快速傅立叶变换

N维快速傅立叶变换用于对高维信号矩阵执行傅立叶频谱分析操作。其中二维的快速傅立叶变换常常用于数字图像处理。

N 维快速傅立叶变换是由一维 FFT 组合而成的，其运算实质就是在给定二维或多维数组的每个维度上依次执行一维FFT，并且使用“原位运算”的方法。在开始之前，算法将输入直接复制到输出上，所以之后在每个维度上执行 FFT 的原位操作都不会改变原本的输入数组，同时也使这个算法输出的数组和输入的数组拥有同样的大小和维度。也就是说，如果对一幅灰度图像执行二维快速傅立叶变换操作，得到的结果也将是一个二维数组。

6.3.6 Matlab实现

Matlab中提供了fft2和ifft2函数分别计算二维傅立叶变换和反变换，它们都经过了优化，运算速度非常快；另一个与傅立叶变换密切相关的函数是fftshift，常需要利用它来将傅立叶频谱图中的零频点移动到频谱图的中心位置。

下面分别介绍这3个函数。

1．fft2函数

该函数用于执行二维快速傅立叶操作，因此可以直接用于数字图像处理。调用语法为

Y = fft2(X)

Y = fft2(X,m,n)

参数说明

● X为输入图像。

● m和n分别用于将X的第一和第二维规整到指定的长度。当m和n均为2的整数次幂时，算法的执行速度要比m和n均为素数时更快。

返回值

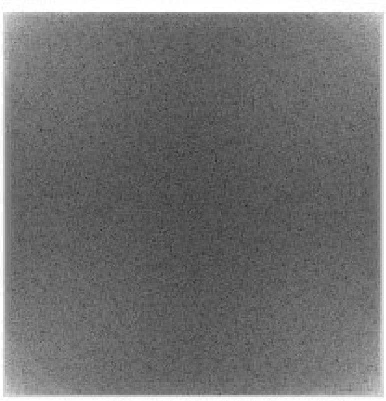
● Y是计算得到的傅立叶频谱，是一个复数矩阵。

提示

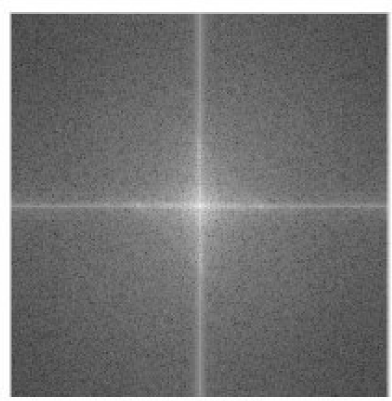
计算abs(Y)可得到幅度谱，计算angle(Y)可得到相位谱。

2．fftshift函数

在fft2函数输出的频谱分析数据中，是按照原始计算所得的顺序来排列频谱的，而没有以零频为中心来排列，因此造成了零频在输频谱矩阵的角上，显示幅度谱图像时表现为4个亮度较高的角（零频处的幅值较高），如图6.11（a）所示。



（a）未经过平移的幅度谱



（b）经过平移的幅度谱

▲图6.11 频谱的平移

fftshift函数利用了频谱的周期性特点，将输出图像的一半平移到另一端，从而使零频被移动到图像的中间。其调用语法为

Y = fftshift(X)

Y = fftshift(X,dim)

参数说明

● X为要平移的频谱。

● dim指出了在多维数组的哪个维度上执行平移操作。

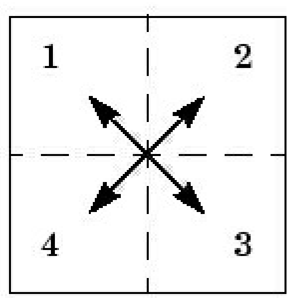
返回值

● Y是经过平移的频谱。

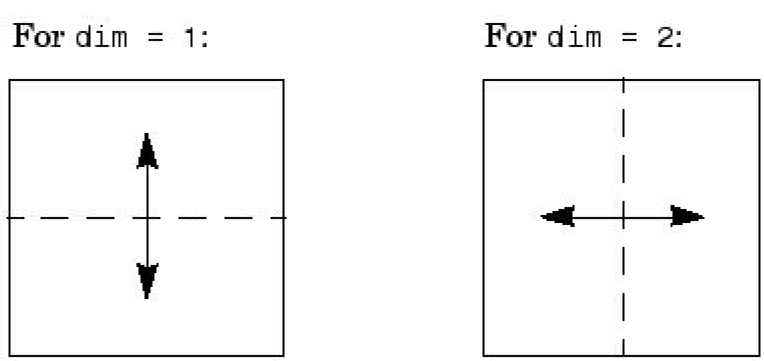
利用fftshift函数对图6.11（a）中的图像平移后的效果如图6.11（b）所示。

下面给出对于二维图像矩阵，fftshift函数的平移过程，如图6.12所示。

可见，输出矩阵被分为了4个部分，其中1、3两部分对换，2、4两部分对换。这样，原来在角上的零频率点（原点）位置就移动到了图像的中央位置。而dim参数则可以指定在多维数组的哪个维度上执行对换操作。例如，对于矩阵而言，dim取1和2的情形分别如图6.13所示。



▲图6.12 FFTSHIFT对二维矩阵的平移过程



▲图6.13 FFTSHIIFT对二维矩阵的平移过程细节分析

3．ifft2函数

该函数用于对图像（矩阵）执行逆傅立叶变换。输出矩阵的大小与输入矩阵相同。调用形式为

Y = ifft2(X)

Y = ifft2(X,m,n)

参数说明

● X为要计算反变换的频谱。

● m、n的意义与fft2中相同。

返回值

● Y是反变换后得到的原始图像。

注意

在执行IFFT2函数之前，如果曾经使用FFTSHIFT函数对频域图像进行过原点平移，则还需要使用IFFTSHIFT将原点平移回原位置。

[例6.1] 幅度谱的意义

下面的程序展示了如何利用fft2进行二维快速傅立叶变换。为了更好地显示频谱图像，需要利用3.3节中学习过的对数变换来增强频谱。

```
I1 = imread('cell.tif');    %读入原图像
fcoef = fft2(double(I1));    %做fft变换
spectrum = fftshift(fcoef);    %将零点移到中心
temp =log(1+abs(spectrum));    %对幅值做对数变换以压缩动态范围
subplot(1,2,1);
imshow(temp,[]);
title('FFT');
subplot(1,2,2);
imshow(I1);
```

```
title('Source')
I2 = imread('circuit.tif');    %读入原图像
fcoef = fft2(double(I2));    %做fft变换
spectrum = fftshift(fcoef);    %将零点移到中心
temp =log(1+abs(spectrum));    %对幅值做对数变换以压缩动态范围
figure;
subplot(1,2,1);
imshow(temp,[]);
title('FFT');
subplot(1,2,2);
imshow(I2);
title('Source')
```

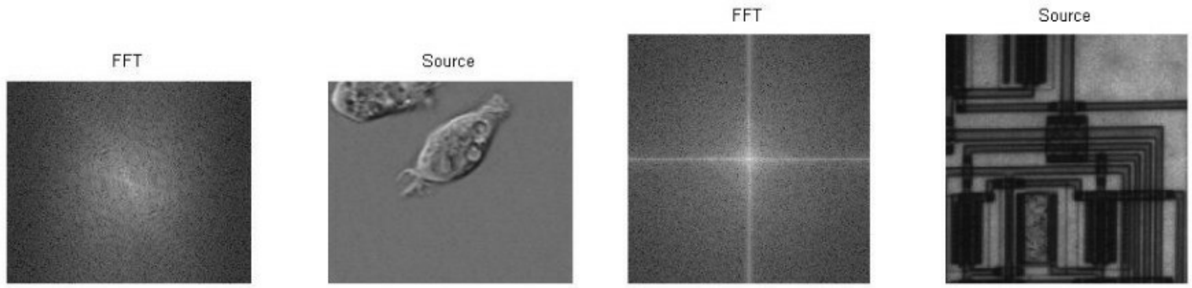
上述程序的运行结果如下。

由图6.14所示可以看出，图6.14（b）中的cell.tif图像较为平滑，而在其傅立叶频谱中，低频部分对应的幅值较大；而对图6.14（d）中细节复杂的的图像circuit.tif，灰度的变化趋势更加剧烈，相应的频谱中高频分量较强。

事实上，由于图6.14（b）图中基本只存在水平和垂直的线条，导致了在输出的频谱中亮线集中存在于水平和垂直方向（并且经过原点）。具体地说，原图像中的水平边缘对应于频谱中的竖直亮线，而竖直边缘则对应着频谱中的水平响应。我们不妨这样理解，水平方向的边缘可以看作在竖直方向上的灰度值的矩形脉冲，而这样的矩形脉冲可以分解为无数个竖直方向正弦平面波的叠加，从而对应频域图像中的垂直亮线；而对于竖直方向的边缘，情况是类似的。

通过例6.1可以发现一些频谱与其空间域图像之间的联系。实际上，低频（频谱图像中靠近中心的区域）对应着图像的慢变化分量；高频（频谱图像中远离中心的区域）对应着一幅图像中较快变化的灰度

级，常常对应着图像细节，如物体的边缘和噪声等。就拿图6.14（c）的电路图像来说，电路板的灰度较为一致的背景区域就对应着频谱的低频部分，而横竖电路线条的灰度变换则是相对高频的成份，且灰度变换越剧烈，就对应着越高的频域分量。



（a） （b） （c） （d）

▲图6.14 图像及其幅度谱

我们在6.2.3小节曾给出了幅度谱和相位谱的定义，并对其作用进行了简单的介绍。为了进一步加深读者对幅度谱和相位谱的认识，这里给出一个关于它们的有趣的例子。

[例6.2]美女与猫——交换两幅图像的相位谱

图6.15（a）、（b）中分别是一张美女的照片和一张猫的照片，这里我们准备交换这两幅图像的相位谱，即用美女的幅度谱加上猫的相位谱，而用猫的幅度谱加上美女的相位谱，然后根据式（6-18），通过幅度谱和相位谱来还原傅立叶变换F(u,v)，再经傅立叶反变换得到交叉相位谱之后的图像。根据6.2.2中关于幅度谱和相位谱各自作用的讨论，您能想到这样做将会产生怎样的结果吗?

```
% ex6_2.m
% 读取图片
A = imread('../beauty.jpg');
B = imread('../cat.jpg');
% 求傅立叶变换
```

```
Af = fft2(double(A));
Bf = fft2(double(B));
% 分别求幅度谱和相位谱
AfA = abs(Af);
AfB = angle(Af);
BfA = abs(Bf);
BfB = angle(Bf);
% 交换相位谱并重建复数矩阵
AfR = AfA .* cos(BfB) + AfA .* sin(BfB) .* i;
BfR = BfA .* cos(AfB) + BfA .* sin(AfB) .* i;
% 傅立叶反变换
AR = abs(ifft2(AfR));
BR = abs(ifft2(BfR));
% 显示图像
subplot(2,2,1);
imshow(A);
title('美女原图像');
subplot(2,2,2);
imshow(B);
title('猫的原图像');
subplot(2,2,3);
imshow(AR, []);
title('美女的幅度谱和猫的相位谱组合');
subplot(2,2,4);
imshow(BR, []);
title('猫的幅度谱和美女的相位谱组合');
```

程序运行结果如图6.15所示。



（a）美女图像



（b）猫的图像



（c）美女幅度谱加上猫的相位谱



（d）猫的幅度谱加上美女的相位谱

▲图6.15 幅度谱与相位谱的关系

通过这个示例，我们可以发现，交换相位谱之后，反变换之后得到的图像内容与其相位谱对应的图像一致，这就印证了我们之前关于相位谱决定图像结构的论断。而图像中整体灰度分布的特性，如明暗、灰度变化趋势等则在比较大的程度上取决于对应的幅度谱，因为幅度谱反映了图像整体上各个方向的频率分量的相对强度。

6.4 频域滤波基础

6.4.1 频域滤波与空域滤波的关系

傅立叶变换可以将图像从空域变换到频域，而傅立叶反变换则可以将图像的频谱逆变换为空域图像，即人可以直接识别的图像。这样一

来，我们可以利用空域图像与频谱之间的对应关系，尝试将空域卷积滤波变换为频域滤波，而后再将频域滤波处理后的图像反变换回空间域，从而达到图像增强的目的。这样做的一个最主要的吸引力在于频域滤波的直观性特点，关于这一点稍后将进行详细的阐述。

根据著名的卷积定理：两个二维连续函数在空间域中的卷积可由其相应的两个傅立叶变换乘积的反变换而得到；反之，在频域中的卷积可由在空间域中乘积的傅立叶变换而得到，即

$$f(x,y)*h(x,y) <==> F(u,v)H(u,v) \quad （6\text{-}47）$$

$$f(x,y)h(x,y) <==> F(u,v)*H(u,v) \quad （6\text{-}48）$$

其中，F(u, v)和H(u, v)分别表示f(x, y)和h(x, y)的傅立叶变换，而符号<==>表示傅立叶变换对，即左侧的表达式可通过傅立叶正变换得到右侧的表达式，而右侧的表达式可通过傅立叶反变换得到左侧的表达式。

式（6-47）构成了整个频域滤波的基础，卷积的概念我们曾在第5章空间域滤波中讨论过，而式中的乘积实际上就是两个二维矩阵F(u, v)和H(u, v)对应元素之间的乘积。

6.4.2 频域滤波的基本步骤

根据式（6-47）进行频域滤波通常应遵循以下步骤。

（1）计算原始图像f(x, y)的DFT，得到F(u, v)。

（2）将频谱F(u, v)的零频点移动到频谱图的中心位置。

（3）计算滤波器函数H(u, v)与F(u, v)的乘积G(u, v)。

（4）将频谱G(u, v)的零频点移回到频谱图的左上角位置。

（5）计算第（4）步计算结果的傅立叶反变换g(x, y)。

（6）取g(x, y)的实部作为最终滤波后的结果图像。

由上面的叙述易知，滤波能否取得理想结果的关键取决于频域滤波函数 H(u, v)。我们常常称之为滤波器，或滤波器传递函数，因为它在滤波中抑制或滤除了频谱中某些频率的分量，而保留其他的一些频率不受影响。本书中我们只关心其值为实数的滤波器，这样滤波过程中 H 的每一个实数元素分别乘以F中对于位置的复数元素，从而使F中元素的实部和虚部等比例的变化，不会改变 F 的相位谱，这种滤波器也因此被称为“零相移”滤波器。这样，最终反变换回空域得到的滤波结果图像g(x, y)理论上也应当为实函数。然而由于计算舍入误差等原因，可能会带有非常小的虚部，通常将虚部直接忽略。

为了更为直观地理解频域滤波与空域滤波之间的对应关系，让我们先来看一个简单的例子。6.2节中曾指出了原点处的傅立叶变换F(0, 0)实际上是图像中全部像素的灰度之和。那么如果我们要从原图像f(x, y)得到一幅像素灰度和为0的空域图像g(x, y)，就可以先将f(x, y)变换到频域F(u, v)，而后令F(0, 0)=0（在原点移动到中心的频谱中为F(M/2, N/2)），再反变换回去。这个滤波过程相当于计算F(u, v)和如下的H(u, v)之间的乘积。

$$H(u,v)=\begin{cases}0, & (u,v)=(M/2,N/2)\\ 1, & \text{其他}\end{cases} \tag{6-49}$$

上式中的H(u, v)对应于平移过的频谱，其原点位于（M/2, N/2）。显然，这里H(u, v)的作用就是将点F(M/2, N/2)置零，而其他位置的F(u, v)保持不变。有兴趣的读者可以自己尝试这个简单的频域滤波过程，反变换之后验证g(x, y)的所有像素灰度之和是否为0。我们将在6.4.2小节详细地探讨一些具有更高实用性的频域滤波器。

6.4.3 频域滤波的Matlab实现

为方便读者在Matlab中进行频域滤波，我们编写了imfreqfilt函数，其用法同空域滤波时使用的imfilter函数类似，调用时需要提供原始图像和与原图像等大的频域滤波器作为参数，函数的输出为经过滤波处理又反变换回空域之后的图像。

注意

通常使用fftshift函数将频谱原点移至图像中心，因此需要构造对应的原点在中心的滤波器，并在滤波之后使用ifftshift函数将原点移回以进行反变换。

频域滤波算法 imfreqfilt 的完整实现如下，该算法被封装在金羽图书论坛（ http://bbs.book 95.com）的“金羽图书与答疑”板块与本书同名的主题帖子附件的“chapter6/code”目录下的imfreqfilt.m文件中。

```
function out = imfreqfilt(I, ff)
% imfreqfilt函数    对灰度图像进行频域滤波
% 参数I    输入的空域图像
% 参数ff    应用的与原图像等大的频域滤镜
if (ndims(I)==3) && (size(I,3)==3) % RGB图像
   I = rgb2gray(I);
end
if (size(I) ～= size(ff))
   msg1 = sprintf('%s: 滤镜与原图像不等大，检查输入', mfilename);
   msg2 = sprintf('%s: 滤波操作已经取消', mfilename);
   eid = sprintf('Images:%s:ImageSizeNotEqual',mfilename);
   error(eid,'%s %s',msg1,msg2);
end
% 快速傅立叶变换
f = fft2(double(I));
% 移动原点
```

```
s = fftshift(f);
% 应用滤镜及反变换
out = s .* ff; %对应元素相乘实现频域滤波
out = ifftshift(out);
out = ifft2(out);
% 求模值
out = abs(out);
% 归一化以便显示
out = out/max(out(:));
```

6.5 频域低通滤波器

在频谱中，低频主要对应图像在平滑区域的总体灰度级分布，而高频对应图像的细节部分，如边缘和噪声。因此，图像平滑可以通过衰减图像频谱中的高频部分来实现，这就建立了空间域图像平滑与频域低通滤波之间对应关系。

6.5.1 理想低通滤波器及其实现

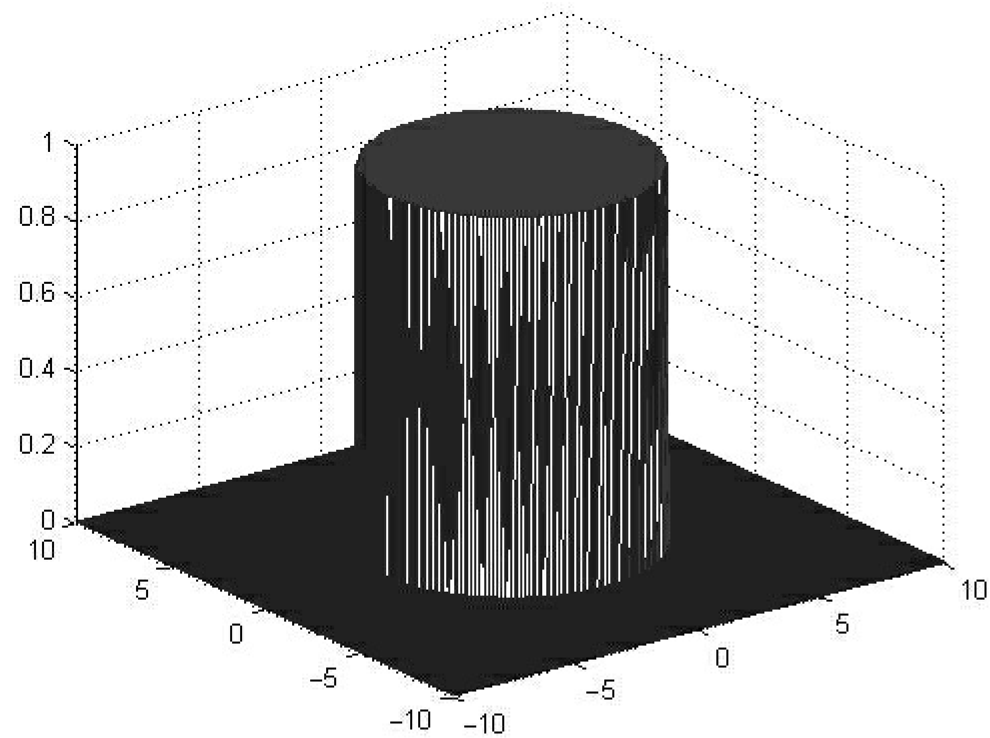
1．理论基础

最容易想到的衰减高频成份的方法就是在一个称为“截止频率”的位置“截断”所有的高频成份，将图像频谱中所有高于这一截止频率的频谱成份设为0，低于截止频率的成份设为保持不变。能够达到这种效果的滤波器如图6.16所示，我们称之为理想低通滤波器。如果图像的宽度为M，高度为N，那么理想低通频域滤波器可形式化地描述为

$$H(u,v)=\begin{cases}1,[(u-\frac{M}{2})^2+(v-\frac{N}{2})^2]\leqslant D_0\\0,[(u-\frac{M}{2})^2+(v-\frac{N}{2})^2]>D_0\end{cases}\qquad（6\text{-}50）$$

其中D_0表示理想低通滤波器的截止频率，滤波器的频率域原点在频谱图像的中心处，在以截止频率为半径的圆形区域之内的滤镜元素值全部为1，而该圆之外的滤镜元素值全部为0。理想低通滤波器的频率特性在截止频率处十分陡峭，无法用硬件实现，这也是我们称之为理想的原因，但其软件编程的模拟实现较为简单。

理想低通滤波器可在一定程度上去除图像噪声，但由此带来的图像边缘和细节的模糊效应也较为明显，其滤波之后的处理效果比较类似于5.3.1中的平均平滑。实际上，理想低通滤波器是一个与频谱图像同样尺寸的二维矩阵，通过将矩阵中对应较高频率的部分设为0，较低频率的部分（靠近中心）设为1，可在与频谱图像相乘后有效去除频谱的高频成份（由于是矩阵对应元素相乘，频谱高频成份与滤波器中的0相乘）。其中0与1的交界处即对应滤波器的截止频率。



▲图6.16 理想低通滤波器曲面图

2．Matlab实现

利用我们编写的imidealflpf函数可以得到截止频率为freq的理想低通滤波器。

imidealflpf 函数的完整实现如下，该函数被封装在金羽图书论坛（http://bbs.book95.com）的“金羽图书与答疑”板块与本书同名的主题帖子附件的“chapter6/code”目录下的imidealflpf.m文件中。

```
function out = imidealflpf(I, freq)
% imidealflpf函数    构造理想的频域低通滤波器
% I参数    输入的灰度图像
% freq参数    低通滤波器的截止频率
% 返回值：out – 指定的理想低通滤波器
[M,N] = size(I);
```

```
out = ones(M,N);
for i=1:M
    for j=1:N
     if (sqrt(((i-M/2)^2+(j-N/2)^2))>freq)
       out(i,j)=0;
     end
    end
end
```

下面我们仍以第 5 章中图像平滑中曾使用过的婴儿老照片 baby_noise.bmp 为例，使用频域理想低通滤波器进行处理，相应的 Matlab代码如例6.3所示。

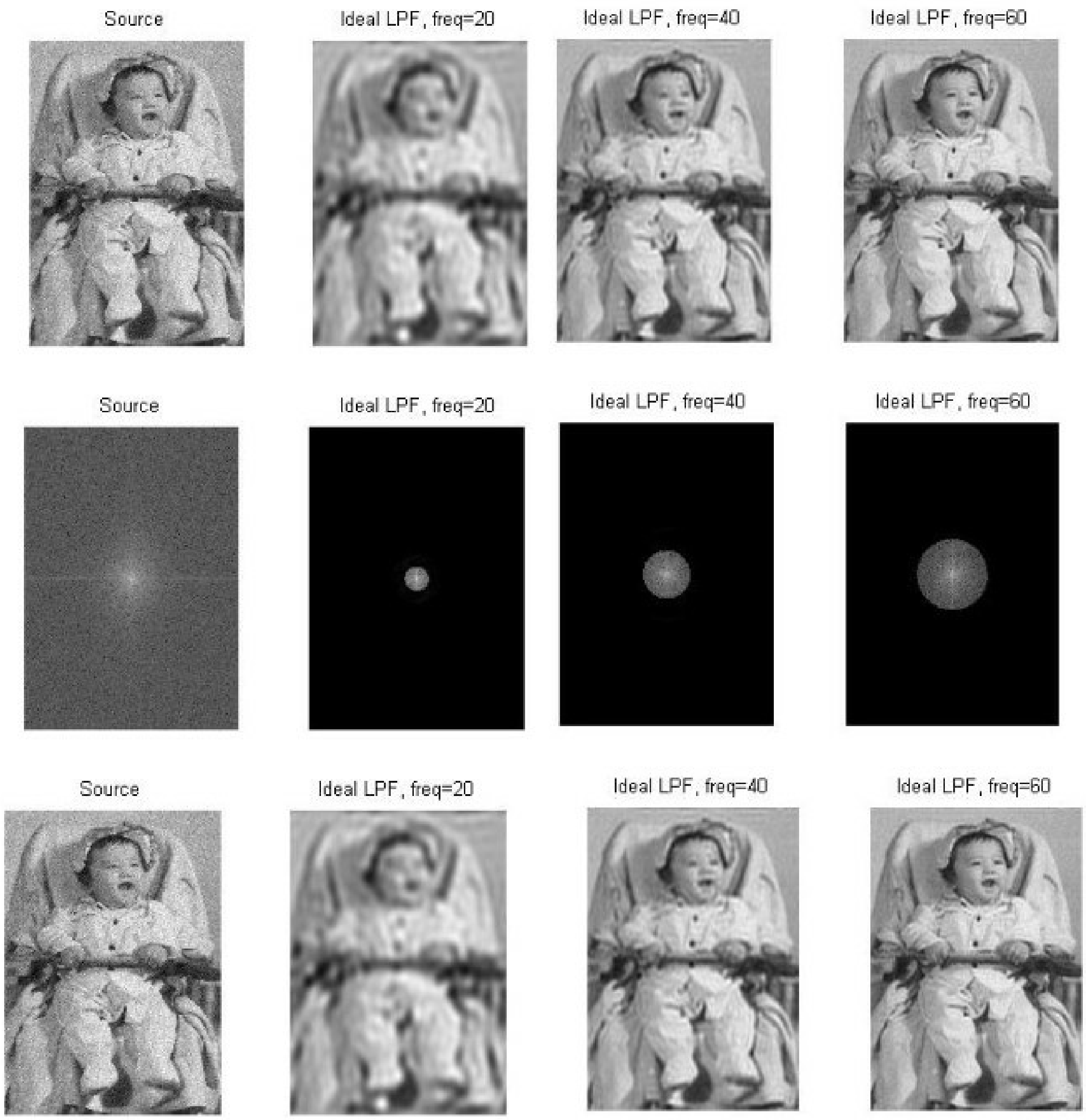
[例6.3]理想低通滤波实现。

```
I = imread('../baby_noise.bmp'); %读入原图像
% 生成滤镜
ff = imidealflpf(I, 20);
% 应用滤镜
out = imfreqfilt(I, ff);
figure (1);
subplot(2,2,1);
imshow(I);
title('Source');
% 计算FFT并显示
temp = fft2(double(I));
temp = fftshift(temp);
temp = log(1 + abs(temp));
figure (2);
subplot(2,2,1);
```

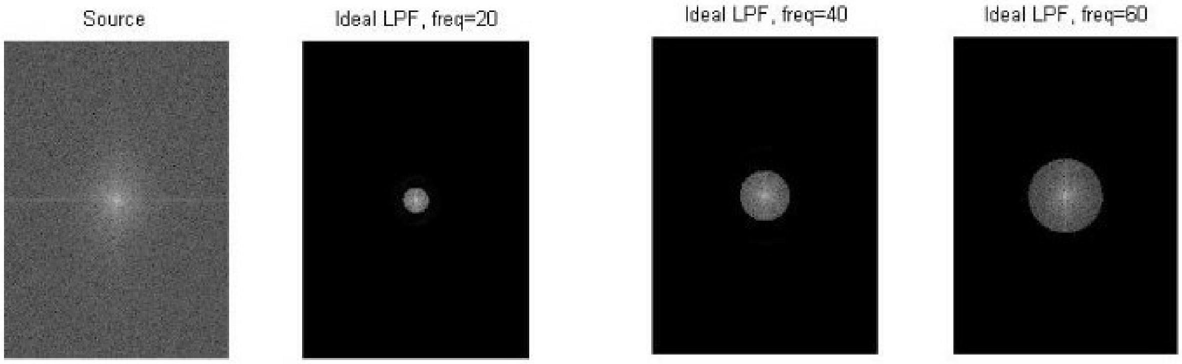
```
imshow(temp, []);
title('Source');
figure (1);
subplot(2,2,2);
imshow(out);
title('Ideal LPF, freq=20');
% 计算FFT并显示
temp = fft2(out);
temp = fftshift(temp);
temp = log(1 + abs(temp));
figure (2);
subplot(2,2,2);
imshow(temp, []);
title(' Ideal LPF, freq=20');
% 生成滤镜
ff = imidealflpf(I, 40);
% 应用滤镜
out = imfreqfilt(I, ff);
figure (1);
subplot(2,2,3);
imshow(out);
title('Ideal LPF, freq=40');
% 计算FFT并显示
temp = fft2(out);
temp = fftshift(temp);
temp = log(1 + abs(temp));
figure (2);
```

```
subplot(2,2,3);
imshow(temp, []);
title(' Ideal LPF, freq=40');
% 生成滤镜
ff = imidealflpf(I, 60);
% 应用滤镜
out = imfreqfilt(I, ff);
figure (1);
subplot(2,2,4);
imshow(out);
title('Ideal LPF, freq=60');
% 计算FFT并显示
temp = fft2(out);
temp = fftshift(temp);
temp = log(1 + abs(temp));
figure (2);
subplot(2,2,4);
imshow(temp, []);
title(' Ideal LPF, freq=60');
```

上述程序的运行效果如图6.17和图6.18所示。



▲图6.17 理想低通滤镜的滤波结果对比图



▲图6.18 理想低通滤镜的滤波结果频谱对比图

从图6.17和图6.18可见，当截止频率非常低时，只有非常靠近原点的低频成份能够通过，图像模糊严重；截止频率越高，通过的频率成份就越多，图像模糊的程度越小，所获得的图像也就越接近原图像。但可以看出，理想低通滤波器并不能很好地兼顾噪声滤除与细节保留两个方面，这和空域中采用平均模板时的情形比较类似。下面，我们将介绍频域的高斯低通滤波器，并比较它与理想低通滤波器的处理效果。

6.5.2 高斯低通滤波器及其实现

1．理论基础

高斯低通滤波器的频率域二维形式由下式给出。

$$H(u,v)=\mathrm{e}^{-[(u-\frac{M}{2})^2+(v-\frac{N}{2})^2]/2\sigma^2} \qquad （6\text{-}51）$$

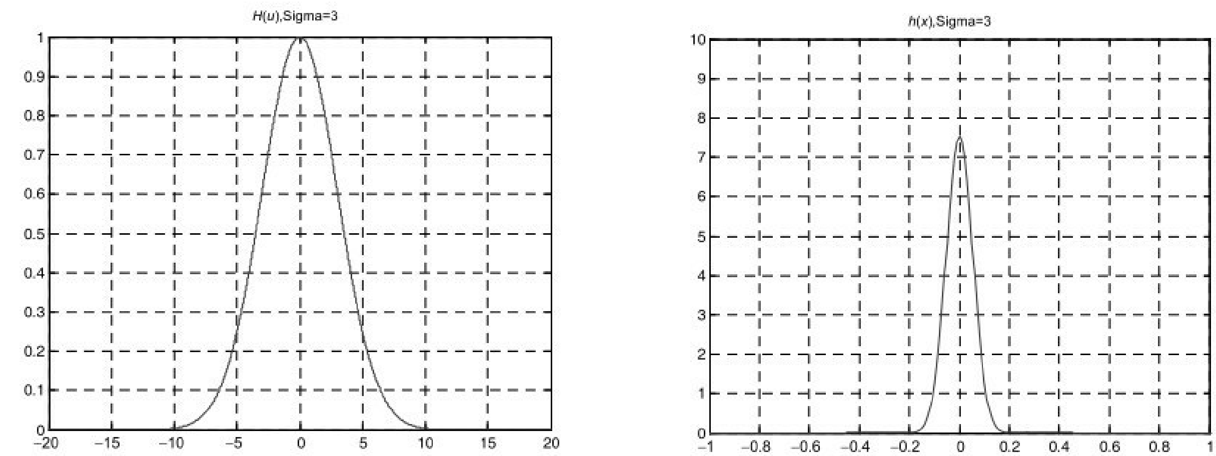
高斯函数具有相对简单的形式，而且它的傅立叶变换和傅立叶反变换都是实高斯函数。为了简单起见，下面仅给出一个一维高斯函数的傅立叶变换和傅立叶反变换作为例子，式（6-52）告诉我们一维高斯函数的傅立叶反变换和正变换仍为高斯函数，该式的证明留给有兴趣的读者自己完成（提示：可以利用高斯分布的概率密度函数在定义域上积分为

1的性质）。

$$H(u)=A\mathrm{e}^{-\frac{u^2}{2\sigma^2}}\xleftrightarrow{FFT}h(x)=\sqrt{2\pi}\sigma A^{-2\pi^2\sigma^2x^2} \quad (6\text{-}52)$$

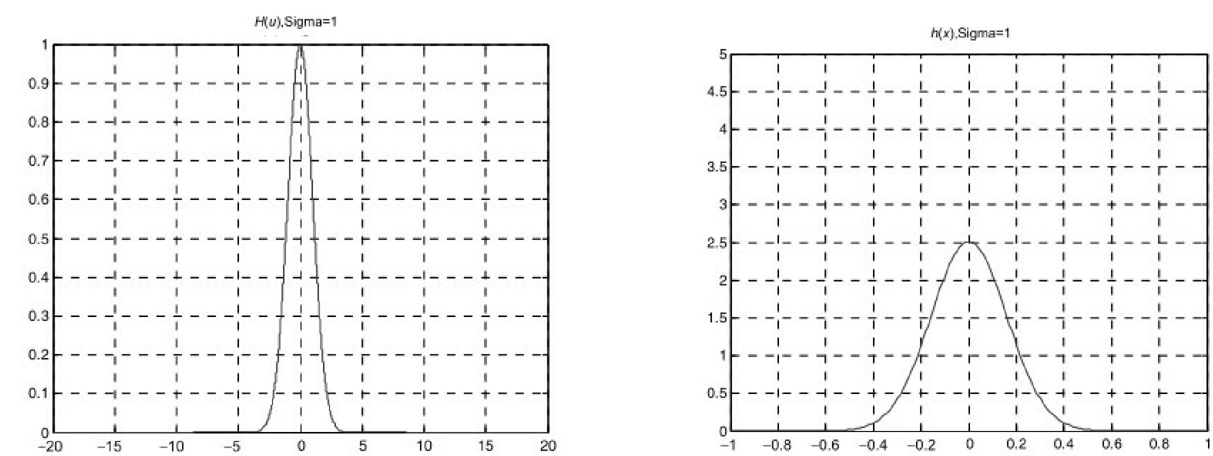
式中：σ是高斯曲线的标准差。

频域和与之对应的空域一维高斯函数的图形如图6.19所示。



（a）σ =3 时高斯函数傅立叶变换的图像　　（b）σ =3 时，高斯函数本身的函数图像

▲图6.19 高斯函数与其傅立叶变换的图像

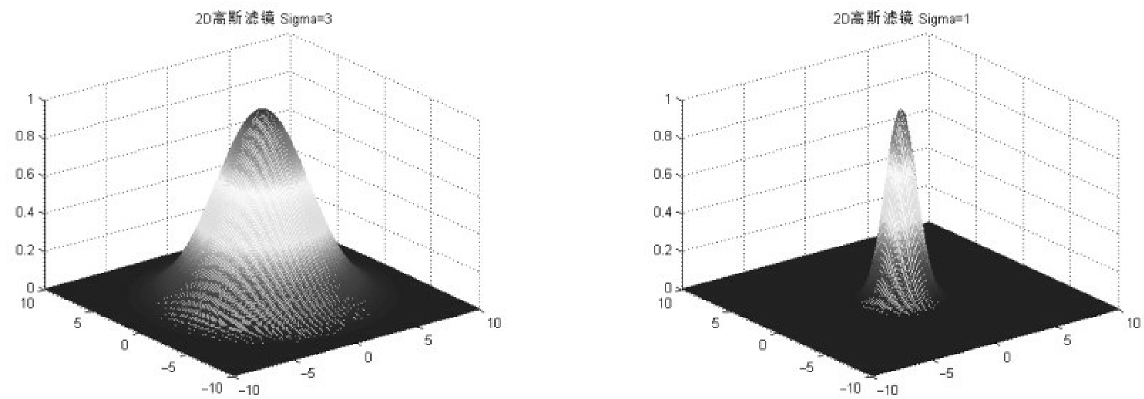


（c）σ =1 时高斯函数傅立叶变换的图像　　（d）σ =1 时，高斯函数本身的函数图像

▲图6.19 高斯函数与其傅立叶变换的图像（续）

从图6.19可以发现，当σ增大时，H(u)的图像倾向于变宽，而h(x)的图像倾向于变窄和变高。这也体现了频率域和空间域的对应关系。频率域滤波器越窄，滤除的高频成份越多，图像就越平滑（模糊）；而在空间域，对应的滤波器就越宽，相应的卷积模板越平坦，平滑（模糊）效果就越明显。

我们在图6.20中分别给出σ取值为3和1时的频域二维高斯滤波器的三维曲面表示，可以看出频域下的二维高斯滤波器同样具有上述一维情况时的特点。



（a）σ=3　　（b）σ=1

▲图6.20 频域二维高斯滤镜的曲面表示

2．Matlab实现

根据上面二维高斯低通滤波器的定义，可以编写高斯低通滤波器的生成函数如下，该函数被封装在金羽图书论坛（http://bbs.book95.com）的“金羽图书与答疑”板块与本书同名的主题帖子附件的“chapter6/code”目录下的imgaussflpf.m文件中。

```
function out = imgaussflpf(I, sigma)
% imgaussflpf函数　构造频域高斯低通滤波器
% I参数　输入的灰度图像
% sigma参数　高斯函数的Sigma参数
```

```
[M,N] = size(I);
out = ones(M,N);
for i=1:M
   for j=1:N
      out(i,j) = exp(-((i-M/2)^2+(j-N/2)^2)/2/sigma^2);
   end
end
```

下面的例6.4给出了针对图像baby_noise.bmp，sigma不同取值时高斯低通滤波的Matlab程序。

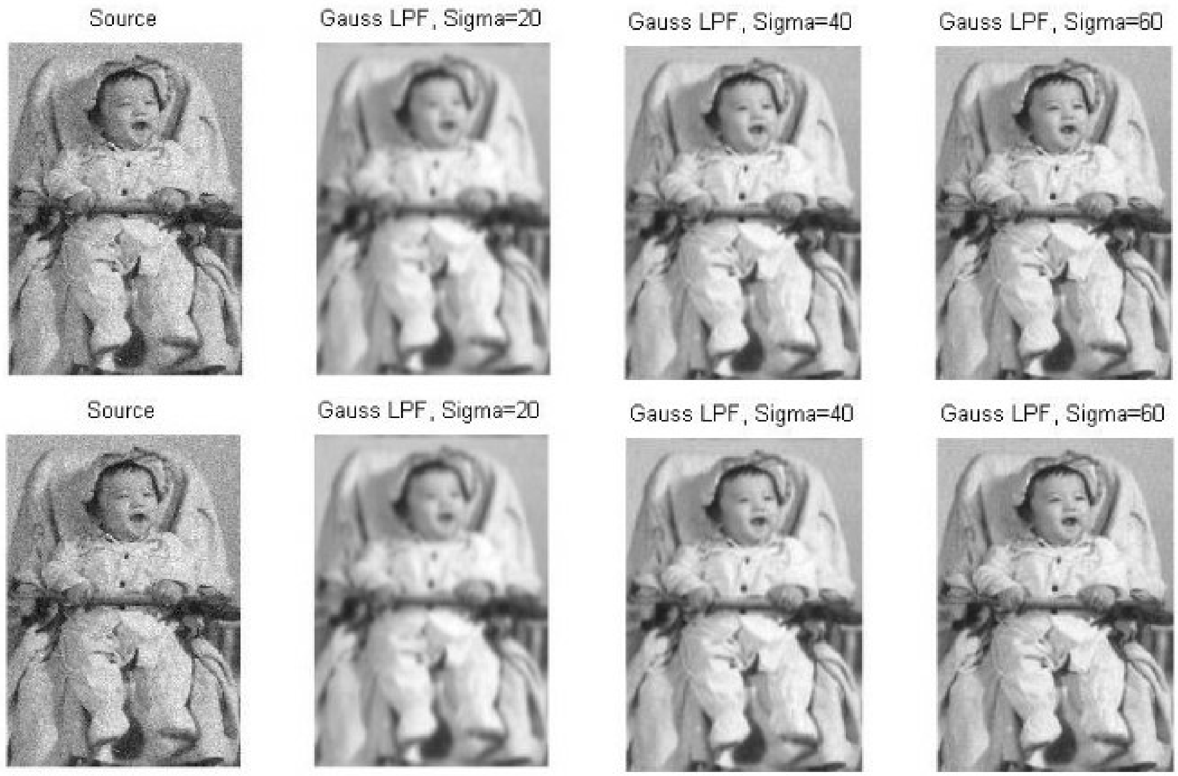
[例6.4]高斯低通滤波实现。

```
I = imread('../baby_noise.bmp'); %读入原图像
% 生成滤镜
ff = imgaussflpf (I, 20);
% 应用滤镜
out = imfreqfilt(I, ff);
figure (1);
subplot(2,2,1);
imshow(I);
title('Source');
% 计算FFT并显示
temp = fft2(double(I));
temp = fftshift(temp);
temp = log(1 + abs(temp));
figure (2);
subplot(2,2,1);
imshow(temp, []);
title('Source');
```

```
figure (1);
subplot(2,2,2);
imshow(out);
title('Gauss LPF, sigma=20');
% 计算FFT并显示
temp = fft2(out);
temp = fftshift(temp);
temp = log(1 + abs(temp));
figure (2);
subplot(2,2,2);
imshow(temp, []);
title(' Gauss LPF, sigma=20');
% 生成滤镜
ff = imgaussflpf (I, 40);
% 应用滤镜
out = imfreqfilt(I, ff);
figure (1);
subplot(2,2,3);
imshow(out);
title('Gauss LPF, sigma =40');
% 计算FFT并显示
temp = fft2(out);
temp = fftshift(temp);
temp = log(1 + abs(temp));
figure (2);
subplot(2,2,3);
imshow(temp, []);
```

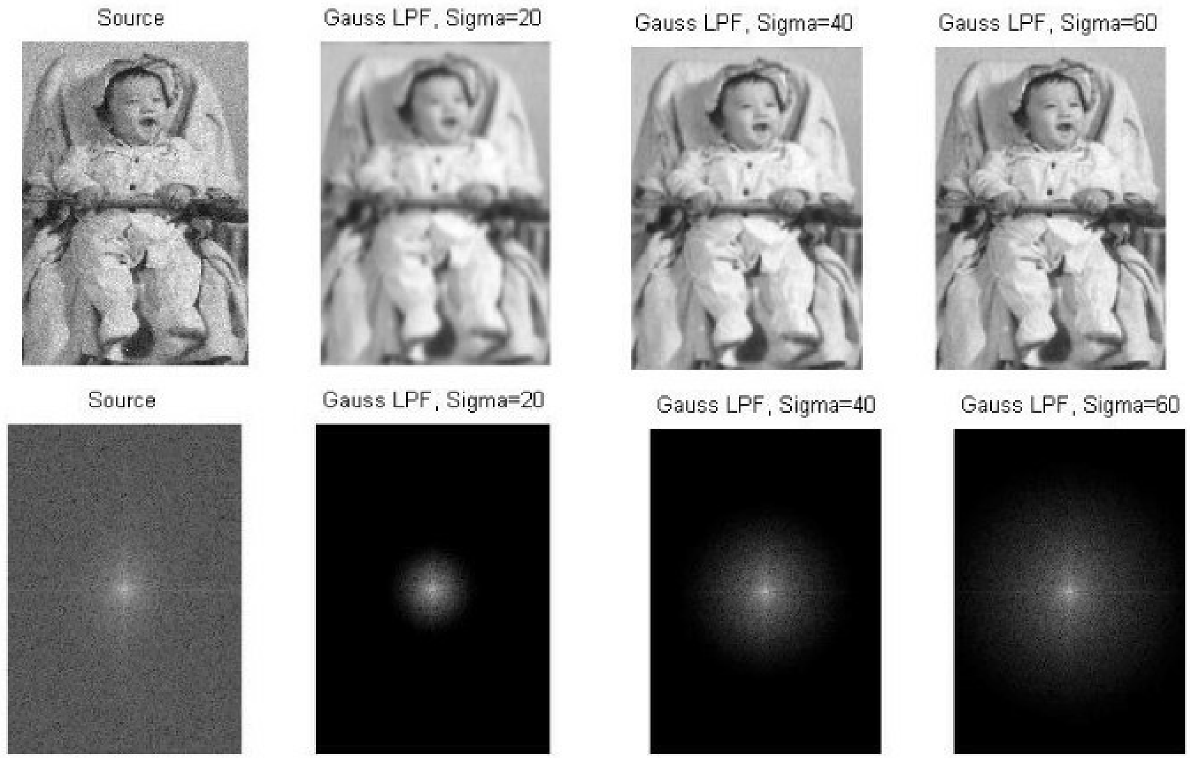
```
title(' Gauss LPF, sigma =40');
% 生成滤镜
ff = imgaussflpf (I, 60);
% 应用滤镜
out = imfreqfilt(I, ff);
figure (1);
subplot(2,2,4);
imshow(out);
title('Gauss LPF, sigma =60');
% 计算FFT并显示
temp = fft2(out);
temp = fftshift(temp);
temp = log(1 + abs(temp));
figure (2);
subplot(2,2,4);
imshow(temp, []);
title(' Gauss LPF, sigma =60');
```

上述程序的运行后得到的滤波效果如图6.21所示。



▲图6.21 高斯低通滤镜的滤波结果对比图

图6.21中各幅图像所对应的频域图像如图6.22所示。显然，高斯滤波器的截止频率处不是陡峭的。



▲图6.22 高斯低通滤镜的滤波结果频域对比图

高斯低通滤波器在Sigma参数取40的时候可以较好地处理被高斯噪声污染的图像，而相比于理想低通滤波器而言，处理效果上的改进是显而易见的。高斯低通滤波器在有效抑制噪声的同时，图像的模糊程度更低，对边缘带来的混叠程度更小，从而使高斯低通滤波器在通常情况下获得了比理想低通滤波器更为广泛的应用。

6.6 频率域高通滤波器

图像锐化可以通过衰减图像频谱中的低频成份来实现，这就建立了空间域图像锐化与频域高通滤波之间对应关系。

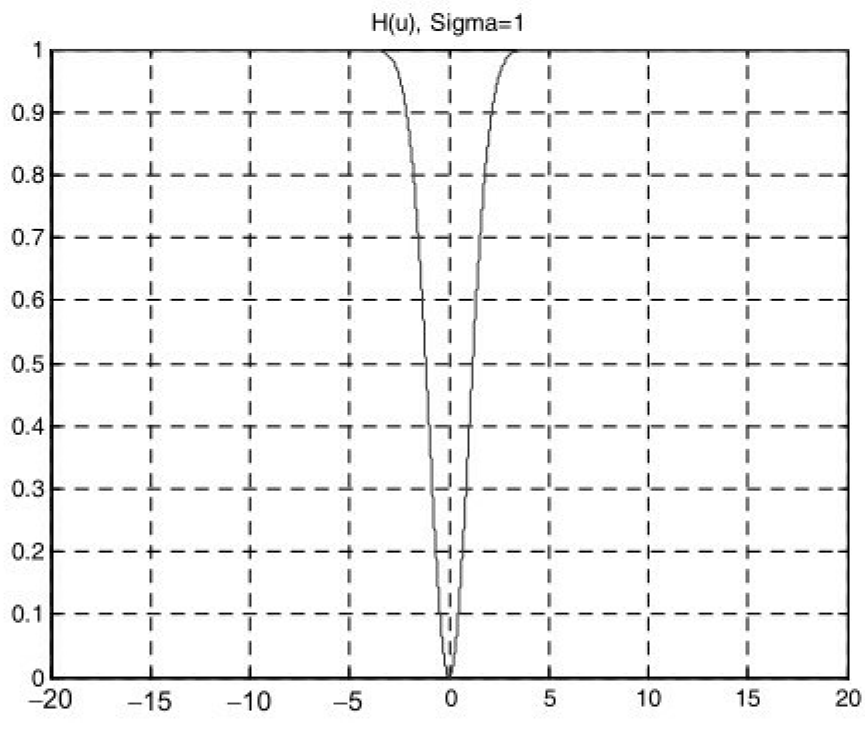
6.6.1 高斯高通滤波器及其实现

1．理论基础

从6.5.2小节高斯低通滤波器中的H(u)图像，可以发现滤波器中心频率处的值即为其最大值1，如果需要做相反的滤波操作，滤除低频成份而留下高频成份，则可以考虑简单地使用如下表达式来获得一个高斯高通滤波器。

$$H(u,v)=1-\mathrm{e}^{-[(u-\frac{M}{2})^2+(v-\frac{N}{2})^2]/2\sigma^2} \quad (6\text{-}53)$$

因此，高斯高通滤波器的频域特性曲线如图6.23所示（仍旧以一维情况为例）。



▲图6.23 高斯高通滤波器的频域特性曲线

2．**Matlab**实现

根据上面二维高斯高通滤波器的定义，可以编写高斯高通滤波器的生成函数如下，该函数被封装在金羽图书论坛（http://bbs.book95.com）

的“金羽图书与答疑”板块与本书同名的主题帖子附件的“chapter6/code”目录下的imgaussfhpf.m文件中。

```
function out = imgaussfhpf(I, sigma)
% imgaussfhpf函数   构造频域高斯高通滤波器
% I参数   输入的灰度图像
% sigma参数   高斯函数的Sigma参数
[M,N] = size(I);
out = ones(M,N);
for i=1：M
  for j=1：N
    out(i,j) = 1 - exp(-((i-M/2)^2+(j-N/2)^2)/2/sigma^2);
  end
end
```

下面的例6.5给出了针对Matlab示例图像coins.png，sigma取不同值时高斯高通滤波的Matlab程序。

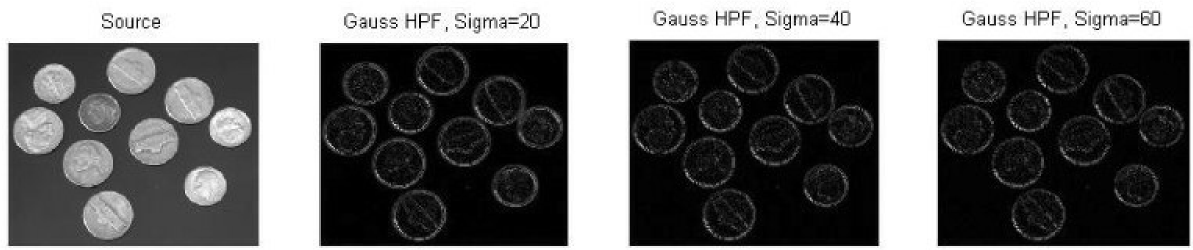
[例6.5]高斯高通滤波实现。

```
I = imread('coins.png');
% 生成滤镜
ff = imgaussfhpf (I, 20);
% 应用滤镜
out = imfreqfilt(I, ff);
figure (1);
subplot(2,2,1);
imshow(I);
title('Source');
% 计算FFT并显示
temp = fft2(double(I));
```

```
temp = fftshift(temp);
temp = log(1 + abs(temp));
figure (2);
subplot(2,2,1);
imshow(temp, []);
title('Source');
figure (1);
subplot(2,2,2);
imshow(out);
title('Gauss HPF, sigma=20');
% 计算FFT并显示
temp = fft2(out);
temp = fftshift(temp);
temp = log(1 + abs(temp));
figure (2);
subplot(2,2,2);
imshow(temp, []);
title('Gauss HPF, sigma=20');
% 生成滤镜
ff = imgaussfhpf (I, 40);
% 应用滤镜
out = imfreqfilt(I, ff);
figure (1);
subplot(2,2,3);
imshow(out);
title('Gauss HPF, sigma =40');
% 计算FFT并显示
```

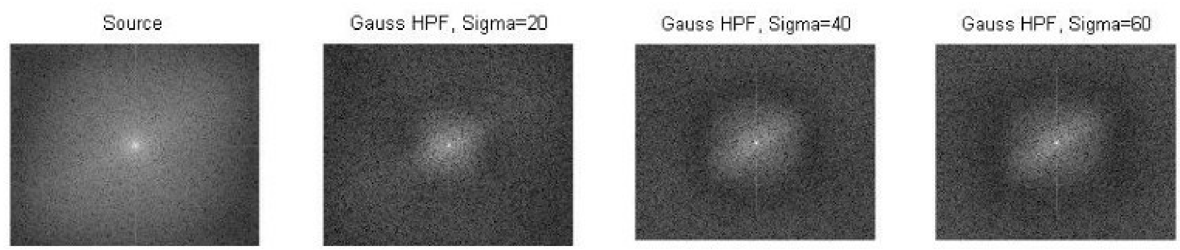
```
temp = fft2(out);
temp = fftshift(temp);
temp = log(1 + abs(temp));
figure (2);
subplot(2,2,3);
imshow(temp, []);
title('Gauss HPF, sigma =40');
% 生成滤镜
ff = imgaussfhpf(I, 60);
% 应用滤镜
out = imfreqfilt(I, ff);
figure (1);
subplot(2,2,4);
imshow(out);
title('Gauss HPF, sigma =60');
% 计算FFT并显示
temp = fft2(out);
temp = fftshift(temp);
temp = log(1 + abs(temp));
figure (2);
subplot(2,2,4);
imshow(temp, []);
title('Gauss HPF, sigma =60');
```

上述程序运行后得到的滤波效果如图6.24所示。



▲图6.24 高斯高通滤镜的滤波结果对比图

滤波前后对应的频域图像如图6.25所示。



▲图6.25 高斯高通滤镜的滤波结果频域对比图

高斯高通滤波器可以较好地提取图像中的边缘信息，Sigma参数取值越小，边缘提取越不精确，会包含越多的非边缘信息；Sigma参数取值越大，边缘提取越精确，但可能包含不完整的边缘信息。

6.6.2 频域拉普拉斯滤波器及其实现

1． 理论基础

频域拉普拉斯算子的推导可以从一维开始，由傅立叶变换的性质可知

$$\mathrm{FFT}[\frac{\mathrm{d}^n f(x)}{\mathrm{d}x^n}] = (\mathrm{j}u)^n F(u) \qquad （6\text{-}54）$$

因此拉普拉斯算子的傅立叶变换计算如下。

$$\mathrm{FFT}[\frac{\partial^2 f(x,y)}{\partial x^2}+\frac{\partial^2 f(x,y)}{\partial y^2}]=(\mathrm{j}u)^2F(u,v)+(\mathrm{j}v)^2F(u,v)=-(u^2+v^2)F(u,v) \qquad (6\text{-}55)$$

因此有下式成立。

$$\mathrm{FFT}[\nabla^2 f(x,y)]=-(u^2+v^2)F(u,v) \qquad (6\text{-}56)$$

也即频域的拉普拉斯滤波器为

$$H(u,v)=-(u^2+v^2) \qquad (6\text{-}57)$$

根据频域图像频率原点的平移规律，将上式改写为

$$H(u,v)=-[(u-\frac{M}{2})^2+(v-\frac{N}{2})^2] \text{（}M\text{和}N\text{分别为图像的宽和高）} \qquad (6\text{-}58)$$

2．Matlab实现

根据式（6-58），可以编写拉普拉斯频域滤波器的生成函数如下，该函数被封装在金羽图书论坛（http://bbs.book95.com）的“金羽图书与答疑”板块与本书同名的主题帖子附件的“chapter6/code”目录下的imlapf.m文件中。

```
function out = imlapf(I)
% imlapf函数    构造频域拉普拉斯滤波器
% I参数    输入的灰度图像
[M,N] = size(I);
out = ones(M,N);
for i=1：M
  for j=1：N
    out(i,j) = -((i-M/2)^2+(j-N/2)^2);
  end
```

```
end
```

下面的例6.6给出了对Matlab示例图像coins.png进行频域拉普拉斯滤波的Matlab程序。

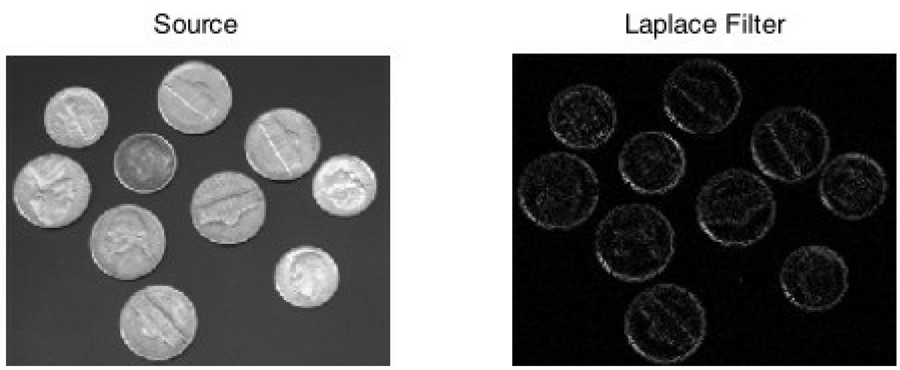
[例6.6] 拉普拉斯滤波实现。

```
I = imread('coins.png');
ff = imlapf (I);
out = imfreqfilt(I, ff);
figure (1);
subplot(1,2,1);
imshow(I);
title('Source');
temp = fft2(double(I));
temp = fftshift(temp);
temp = log(1 + abs(temp));
figure (2);
subplot(1,2,1);
imshow(temp, []);
title('Source');
figure (1);
subplot(1,2,2);
imshow(out);
title('Laplace Filter');
temp = fft2(out);
temp = fftshift(temp);
temp = log(1 + abs(temp));
figure (2);
subplot(1,2,2);
```

imshow(temp, []);

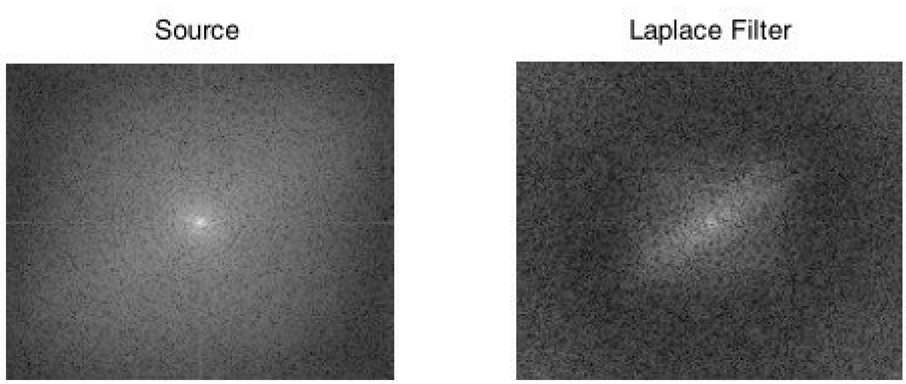
title('Laplace Filter');

上述程序运行后得到的滤波效果如图6.26所示。



▲图6.26 拉普拉斯滤镜的滤波结果对比图

得到的滤波前后的频域图像如图6.27所示。



▲图6.27 拉普拉斯滤镜的滤波结果频域对比图

6.7 Matlab综合案例 —— 利用频域滤波消除周期噪声

6.5～6.6 节介绍了几种典型的频域滤波器，实现了频域下的低通和高通滤波，它们均可在空域下采用平滑和锐化算子实现。而本节准备给出一个特别适合在频域中完成的滤波案例，即利用频域带阻滤波器消除

图像中的周期噪声。下面就来看一下这个在空域中几乎不可能完成的任务是如何在频域中实现的。

6.7.1 频域带阻滤波器

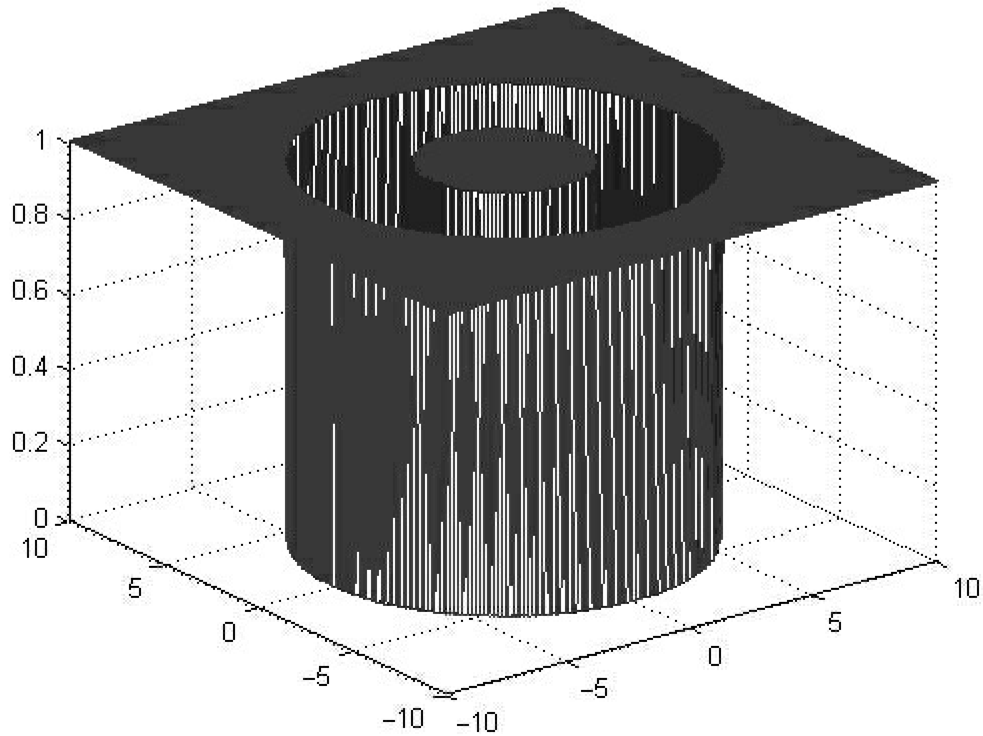
顾名思义，所谓“带阻”就是阻止频谱中某一频带范围的分量通过，其他频率成份则不受影响。常见的带阻滤波器有理想带阻滤波器和高斯带阻滤波器。

1．理想带阻滤波器

理想带阻滤波器的表达式为

$$H(u,v)=\begin{cases}0,|D-D_0|\leqslant W\\1,|D-D_0|>W\end{cases} \tag{6-59}$$

式中：D_0是阻塞频带中心频率到频率原点的距离；W是阻塞频带宽度；D是（u,v）点到频率原点的距离。于是，理想带阻滤波器的频域特性曲面如图6.28所示。



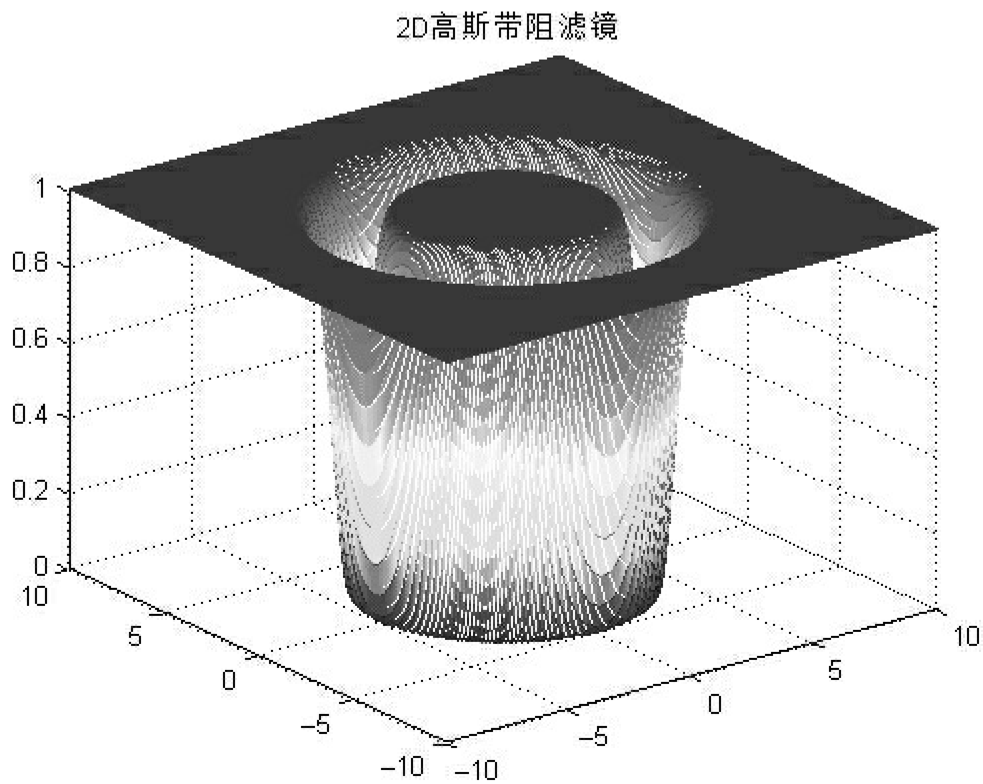
▲图6.28 理想带阻滤波器的频域特性

2．高斯带阻滤波器

本案例中使用了高斯带阻滤波器，下面直接给出高斯带阻滤波器的表达式。

$$H(u,v)=1-\mathrm{e}^{-\frac{1}{2}\left[\frac{D^2(u,v)-{D_0}^2}{D(u,v)W}\right]^2} \qquad (6\text{-}60)$$

式中：D_0是阻塞频带中心频率到频率原点的距离；W是阻塞频带宽度；D是（u,v）点到频率原点的距离。于是，二维高斯带阻滤波器的频域特性曲面如图6.29所示。



▲图6.29 高斯带阻滤波器的频域特性

3．高斯带阻滤波器的**Matlab**实现

根据高斯带阻滤波器的定义式（6-60），可以编写高斯带阻滤波器的生成函数如下，该函数被封装在金羽图书论坛（http://bbs.book95.com）的“金羽图书与答疑”板块与本书同名的主题帖子附件的“chapter6/code”目录下的imgaussfbrf.m文件中。

```
function out = imgaussfbrf(I, freq, width)
% imgaussfbrf函数  构造频域高斯带阻滤波器
% I参数  输入的灰度图像
% freq参数  阻带中心频率
% width参数  阻带宽度
[M,N] = size(I);
out = ones(M,N);
```

```
for i=1：M
  for j=1：N
    out(i,j) = 1-exp(-0.5*((((i-M/2)^2+(j-N/2)^2)-freq^2)/(sqrt(i.^2+j.^2)*width))^2);
  end
end
```

6.7.2 带阻滤波消除周期噪声

带阻滤波器常用于处理含有周期性噪声的图像。周期性噪声可能由多种因素引入，例如图像获取系统中的电子元件等。本案例中我们人为地生成了一幅带有周期噪声的图像，而后通过观察分析其频谱特征，选择了合适的高斯带阻滤波器进行频域滤波。

1．得到周期噪声图像

通常可以使用正弦平面波来描绘周期性噪声。如下程序为Matlab示例图片pout.tif增加周期性噪声。

```
O = imread('pout.tif'); %读入原图像
[M,N] = size(O);
I = O;
for i=1:M
for j=1:N
  I(i,j)=I(i,j)+20*sin(20*i)+20*sin(20*j); %添加周期噪声
end
end
subplot(1,2,1);
imshow(O);
title('Source');
```

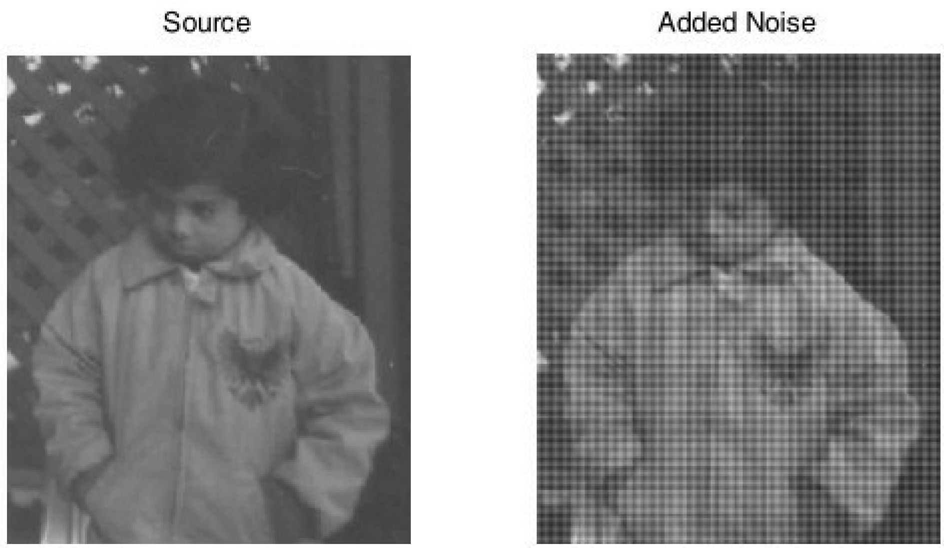
```
subplot(1,2,2);
imshow(I);
title('Added Noise');
```

添加周期性噪声前后的区别如图6.30所示。

2．频谱分析

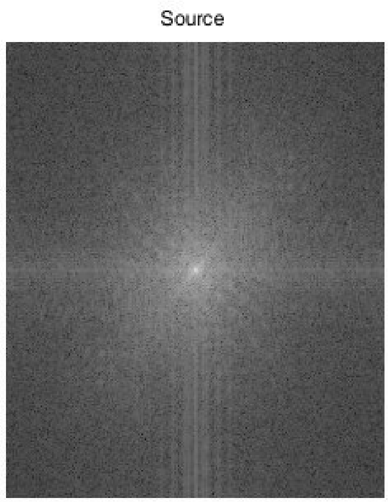
使用高斯带阻滤波器时，首先需要对欲处理的图像的频谱有一个了解。下面的命令得到了两幅图像的频谱。

```
i_f=fft2(I);
i_f=fftshift(i_f);
i_f=abs(i_f);
i_f=log(1+i_f);
o_f=fft2(O);
o_f=fftshift(o_f);
o_f=abs(o_f);
o_f=log(1+o_f);
figure(1);
imshow(o_f, [ ]); %得到图6.31（a）
title('Source');
figure(2);
imshow(i_f, [ ]); %得到图6.31（b）
title('Added Noise');
```

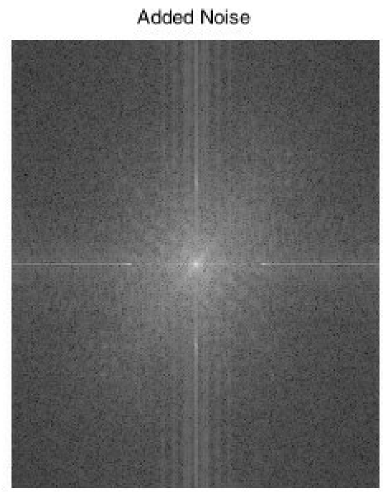


▲图6.30 添加周期噪声前后对比图

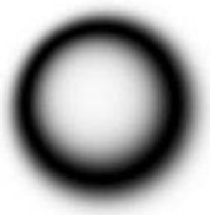
程序的运行结果如图6.31所示。



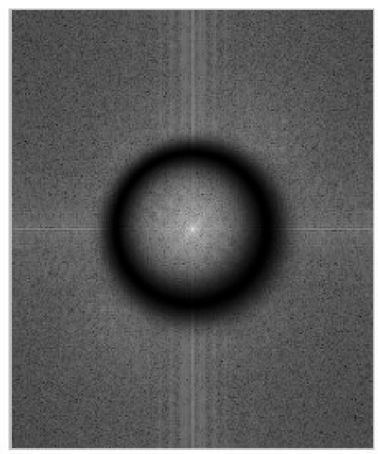
（a）原图像的频谱



（b）加入周期噪声后图像的频谱



（c）高斯带阻滤波器



（d）图（b）经（c）中的滤波器频域滤波（矩阵对应元素相乘）之后

▲图6.31 高斯带阻滤镜的滤波结果频域对比图

3．带阻滤波

观察图6.31（b），发现周期性图像的傅立叶频谱中出现了两对相对于坐标轴对称的亮点，它们分别对应于图像中水平和竖直方向的正弦噪声。我们构造高斯带阻滤波器的时候就需要考虑尽可能滤除具有这些亮点对应的频率的正弦噪声。注意到这4个点位于以频谱原点为中心，以50为半径的圆周上。因此，设置带阻滤波器中心频率为50，频带宽度为5，如图6.31（c）所示，滤波后的频域效果如图6.31（d）所示。

相应的程序如下。

```
ff = imgaussfbrf(I, 50, 5); %构造高斯带阻滤波器
figure, imshow(ff, []); %得到图6.31（c）
out = imfreqfilt(I, ff); %带阻滤波
figure, imshow(out, []); %得到图6.31（d）
subplot(1,2,1);
imshow(I);
title('Source');
subplot(1,2,2);
imshow(out);
title('Gauss Filter');
```

上述程序运行后得到的高斯带阻滤波器最终滤波效果如图 6.32 所示，我们看到周期噪声被很好地消除，这样的效果在空域中是很难实现的。



▲图6.32 高斯带阻滤镜的滤波结果对比图

6.8 频域滤波器与空域滤波器之间的内在联系

在6.4.1小节我们曾探讨了频域滤波与空域滤波之间的关系。这里则要更进一步，来研究频域滤波器与空域滤波器之间的内在联系。

频域滤波较空域而言更为直观，频域下滤波器表达了一系列空域处理（平滑、锐化等）的本质，即对高于/低于某一特定频率的灰度变化信息予以滤除，而对其他的灰度变化信息基本保持不变。这种直观性增加了频域滤波器设计的合理性，使得我们更容易设计出针对特定问题的频域滤波器，就如在6.7节中我们利用了带阻滤波器实现了对图像中周期噪声的滤除，而想直接在空域中设计出一个能够完成如此滤波任务的滤波器（卷积模板）是相当困难的。

为了得到合适的空域滤波器，我们很自然地想到可以首先设计频域滤波器 H(u, v)，而后根据卷积定理式（6-61），将H(u, v)反变换至空域后就得到了空域中滤波使用的卷积模板h(x, y)，从而解决了空域滤波器的设计难题。

然而，直接反变换得到的空域卷积模板h(x, y)同 H(u, v)等大，从而与图像f(x, y)具有相同的尺寸。而模板操作十分耗时，要计算这样大的模板与图像的卷积将是非常低效的。在第3章中我们使用的都是很小的模板（如 3×3, 5×5, 7×7等），因为这样的模板在空域中才具有滤波效率上的优势。一般来说，如果空域模板中的非零元素数目小于132（大约13×13见方），则直接在空域中计算卷积较为划算，否则直接利用H(u, v)在频域下滤波更为合适。

在实际中我们发现，利用以全尺寸的空域滤波器h(x, y)为指导设计出的形状与之类似的小空域卷积模板，同样可以取得类似于频域滤波器H(u, v)的滤波效果。这就为从频域出发，最终设计出具有实用价值的空域模板提供了一种完美的解决方案。

式（6-52）给出的高斯频域低通滤波器 H(u)及与其构成傅立叶变换对儿的空域高斯模板 h(x)正好印证了上述结论。从图6.19上来看，H(u)越窄，h(x)就越宽。而频域低通滤波器H(u)越窄，说明能够通过的频率

越低，被截断的高频成份也就越多，从而使滤波处理后原函数f(x)变得平滑；而空域下以越宽的模板h(x)与函数f(x)卷积则同样会产生平滑的效果。再进一步以h(x)的形状为指导，就可以得到曾在高斯平滑中使用的高斯模板式（5-5）。

附录

Ⅰ.傅立叶级数的收敛性

傅立叶级数的收敛性：满足狄利赫里条件的周期函数表示成的傅里叶级数都收敛。狄利赫里条件如下。

（1）在任何周期内，x(t)须绝对可积。

（2）在任一有限区间中，x(t)只能取有限个最大值或最小值。

（3）在任何有限区间上，x(t)只能有有限个第一类间断点。

Ⅱ.傅立叶级数的三角形式和复指数形式的转换

利用欧拉公式

$$\cos t=\frac{\mathrm{e}^{it}+\mathrm{e}^{-it}}{2},\quad \sin t=\frac{\mathrm{e}^{it}-\mathrm{e}^{-it}}{2i} \qquad (6\text{-}61)$$

式（6-1）可转化为

$$\begin{aligned}&\frac{a_0}{2}+\sum_{n=1}^{+\infty}[\frac{a_n}{2}(\mathrm{e}^{in2\pi ux}+\mathrm{e}^{-in2\pi ux})-\frac{ib_n}{2}(\mathrm{e}^{in2\pi ux}-\mathrm{e}^{-in2\pi ux})]\\&=\frac{a_0}{2}+\sum_{n=1}^{+\infty}[\frac{a_n-ib_n}{2}\mathrm{e}^{in2\pi ux}+\frac{a_n+ib_n}{2}\mathrm{e}^{-in2\pi ux}]\end{aligned} \qquad (6\text{-}62)$$

令$c_0=\frac{a_0}{2},\quad c_n=\frac{a_n-ib_n}{2},\quad c_{-n}=\frac{a_n+ib_n}{2}\qquad (n=1,2,3,\ldots)$　　（6-63）

则式(6-62)可表示为

$$c_0 + \sum_{n=1}^{\infty} c_n \mathrm{e}^{in2\pi ux} + c_{-n}\mathrm{e}^{in2\pi ux}$$

可将上式合并，得到傅立叶级数的复数形式。

$$\sum_{n=-\infty}^{\infty} c_n \mathrm{e}^{in2\pi ux}$$

为求得系数c_n,将式（6-2）代入式（6-63），得

$$c_0 = \frac{a_0}{2} = \frac{1}{T}\int_{-T/2}^{T/2} f(x)\mathrm{d}x$$

$$\begin{aligned} c_n = \frac{a_n - ib_n}{2} &= \frac{1}{2}[\frac{2}{T}\int_{-T/2}^{T/2} f(x)\cos(n2\pi ux)\mathrm{d}x - \frac{2i}{T}\int_{-T/2}^{T/2} f(x)\sin(n2\pi ux)\mathrm{d}x] \\ &= \frac{1}{T}\int_{-T/2}^{T/2} f(x)(\cos(n2\pi ux) - \mathrm{i}\sin(n2\pi ux))\mathrm{d}x \\ &= \frac{1}{T}\int_{-T/2}^{T/2} f(x)\mathrm{e}^{-in2\pi ux}\mathrm{d}x \qquad (n = 1,2,3,\ldots) \end{aligned}$$

$$c_{-n} = \frac{a_n + ib_n}{2} = \frac{1}{T}\int_{-T/2}^{T/2} f(x)\mathrm{e}^{in2\pi ux}\mathrm{d}x \qquad (n = 1,2,3,\ldots)$$

同样的，可将结果合并写为

$$c_n = \frac{1}{T}\int_{-T/2}^{T/2} f(x)\mathrm{e}^{-in2\pi ux}\mathrm{d}x \qquad (n = 0,\pm 1,\pm 2,\pm 3,\ldots)$$

即傅立叶系数的复数形式。

式中: u表示f(x)的频率，即

$$u = \frac{1}{T}，\text{而}\ \omega_0 = \frac{2\pi}{T} = 2\pi u$$

III.傅立叶级数复指数形式和余弦形式间的转换

由复指数形式的傅立叶级数

$$\hat{f}(x)=\sum_{n=-\infty}^{\infty} c_n e^{in2\pi ux}$$

$$=c_0+\sum_{n=1}^{\infty}(c_n e^{in2\pi ux}+c_{-n}e^{i(-n)2\pi ux})$$

∵$c_n=|c_n|e^{i\theta n}$，其中$\theta_n=\arg\tan\frac{\mathrm{Im}(c_n)}{\mathrm{Re}(c_n)}$

即c_n的虚部比c_n的虚部，为n次频率的谐波的相位。

又∵$|c_{-n}|=|c_n|$，而$\theta_{-n}=-\theta_n \Rightarrow c_n=|c_n|e^{-i\theta n}$

$$\therefore \quad \hat{f}(x)=c_0+\sum_{n=1}^{\infty}|c_n|(e^{i(n2\pi ux+\theta_n)}+e^{-i(n2\pi ux+\theta_n)})$$

根据欧拉公式展开2个复指数项易得傅立叶级数展开的复指数形式。

$$\hat{f}(x)=c_0+\sum_{n=1}^{\infty}2|c_n|\cos(n2\pi ux+\theta_n)$$

第7章 小波变换

近年来，随着人们对图像压缩、边缘和特征检测以及纹理分析的需求的提高，一种新的变换（称为小波变换）悄然出现。傅立叶变换一直是变换域图像处理的基石，它能用正弦函数之和表示任何分析函数，而小波变换则基于一些有限宽度的基小波，这些小波不仅在频率上是变化的，而且具有有限的持续时间。比如对于一张乐谱，小波变换不仅能提供要演奏的音符，而且说明了何时演奏等细节信息。但是傅立叶变换只提供了音符，局部信息在变换中丢失。

本章的知识与热点

- 多分辨率框架
- 分解与重构的实现
- Gabor多分辨率分析
- 常见小波分析
- 高维小波

本章的典型案例分析

- 基于Daubechies小波的二维图像分解与重构

7.1 多分辨率分析

多分辨率理论是一种全新而有效的信号处理与分析方法。正如其名字所表达的，多分辨率理论与多种分辨率下的信号（或图像）表示和分析有关。其优势很明显，某种分辨率下无法发现的特性在另一种分辨率下将很容易被发现。本节将从多分辨率的角度来审视小波变换，主要内容将集中在用于简单介绍多分辨率的相关概念和信号（或图像）的分解与重构算法。

7.1.1 多分辨率框架

多分辨率分析又称为多尺度分析，是小波分析中的重要部分，它将多种学科的技术有效地统一在一起，如信号处理的子带编码、数字语音识别的积分镜像过滤以及金字塔图像处理。多分辨率分析的作用是将信号分解成不同空间的部分。另外，它也提供了一种构造小波的统一框架。在观察图像时，对于不同大小的物体，往往采用不同的分辨率，若物体不仅尺寸有大有小，而且对比有强有弱，则采用多分辨率进行分析就凸显出一定的优势。

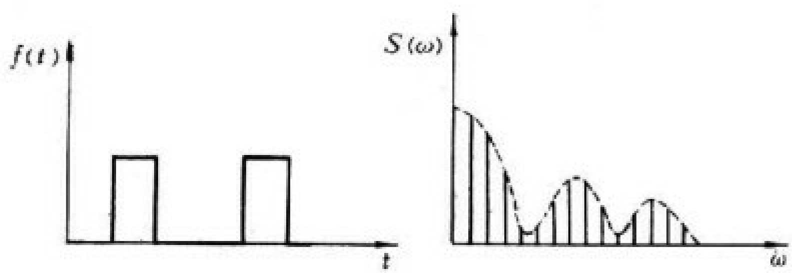
例如，地图通常以不同尺度描得，一幅地图的尺度是地域实际大小与它在地图上的表示的比值。在地球仪上，大陆和海洋等主要特征是可见的，而像城市街道这样的细节信息就很难辨识了；而在较小的尺度上，细节变得可见而较大特征却不见了。因此，为了能够从当地导引到一个较远距离处的地点，就需要一套用不同尺度绘制的地图。如图7.1所示，在中国地图上无法看到详细的北京的街道，而在分辨率较高北京地图上，就能真切地看到北京的各个街道。这就是小波多分辨率的优越性的体现。



（a） 中国地图（部分地图）只供参考用　　（b） 北京街道地图

▲图7.1 中国地图与北京街道地图

小波变换正是沿着多分辨率这条线发展起来的。与时域分析一样，一个信号用一个二维空间表示，不过这里的纵轴是尺度而不是频率，如图7.2所示。根据时频域分析，一个信号的每个瞬态分量映射到时间-频率平面上的位置对应于分量主要频率和发生的时间。
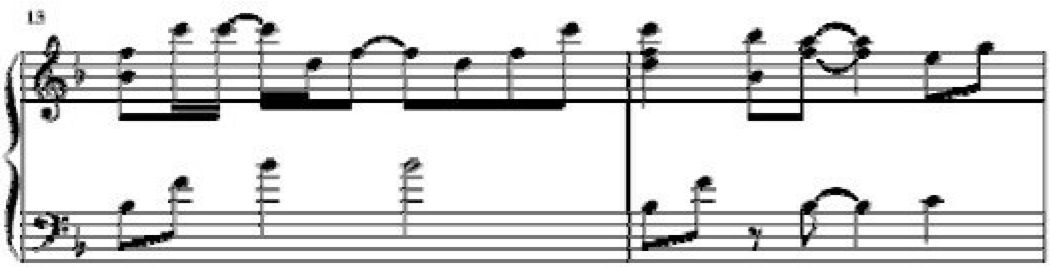


（a）　　（b）

▲图7.2 时域与频域图

考虑图7.3所示的乐谱，它可以看做一个描绘了二维的时频空间。频率（音高）从层次的底部向上增加，而时间（以节拍来测度）则向右

发展。乐章中每一个音符对应于一个将出现在这首歌的演出记录中小波分量（音调猝发）。每一个小波持续宽度都由音符的类型来编码，而不是由它的水平延伸来编码。假设要分析一次音乐演出的记录，并写出相应的乐谱，这个过程可以说是小波变换。同样，音乐家一首歌的演奏录音可看作是一种小波逆变换，因为它是用时频来重构信号的。



▲图7.3 乐谱看出时频图

1988 年，Mallat 提出了多分辨率分析的概念，从空间的概念上形象地说明了小波的多分辨率特性。在介绍多分辨率分析之前，我们先来了解一下关于小波变换的一些基本概念，这对于我们理解多分辨率分析有重要作用。

首先需要介绍一下与小波分析有关的泛函分析的基础知识。泛函分析是20世纪初发展起来的一个重要数学分支，其中一个非常重要的基本概念就是函数空间。函数空间就是由函数构成的集合。常用的函数空间包括距离空间、线性空间、线性赋范空间、巴拿赫空间以及希尔伯特空间，在此我们就不一一详述了，主要以线性空间为例简单说明。

设X为一非空集合，若在X中规定了线性运算，即元素的加法和元素的数乘运算，并满足相应的加法或数乘的结合律及分配律，则称 X 为一线性空间或向量空间。对于线性空间的任一向量我们用范数来定义其长度。

介绍了函数空间的概念后，下面我们介绍函数空间的基及函数展开

的问题。在构造小波函数和进行小波变换的分析、处理过程中，我们会遇到正交基概念问题。所以需要介绍这些重要概念。

所谓基，就是由函数序列组成的空间。设$e_k(t)$为一函数序列，X表示$e_k(t)$所有可能的线性组合构成的集合，即

$$X=\{\sum_k a_k e_k(t) \mid t, a_k \in R, k \in Z\} \qquad (7\text{-}1)$$

称X为序列$e_k(t)$组成的线性空间，即X=span$\{e_k\}$，也即对任意g(t)，可以表达为

$$g(t)=\sum_k a_k e_k(t) \qquad (7\text{-}2)$$

如果 $e_k(t)$线性无关，对任意 g(t)，式（7-2）中的系数 a_k取唯一的值，这时，我们称$\{e_k(t)\}_k \in Z$为空间X的一个基底。

设x,y为内积空间X的两个元素，若(x,y)=0，则称x,y正交。若X中的元素列$\{e_k\}$满足

$$\langle e_m, e_n \rangle = \begin{cases} 0 & m \neq n \\ 1 & m = n \end{cases} \qquad (7\text{-}3)$$

则称$\{e_k\}$为X中的标准正交系。

在小波分析中，多分辨率分析（我们在以后将会详细提到）的核心就是v_j,W_j空间的正交归一基$\varphi_{jk}(t)$,$\phi_{jk}(t)$,k∈z,j∈z（k为整数位移，j为分辨率的级数）。只要它们已知，就可以将待分析函数x(t)投影到不同分辨率的函数空间进行分析。在了解了这些概念后，我们引入小波的概念。

在小波分析中，主要讨论的函数空间为$L^2(R)$。$L^2(R)$指R上平方可积函数构成的函数空间，即

$$f(t) \in L^2(R) \Leftrightarrow \int_R |f(t)|^2 \, \mathrm{d}t < +\infty$$

若 f (t)∈L^2 (R)，则称 f (t)为能量有限的信号。L^2 (R)也常称为能量有限的信号空间。

令φ(t)∈L^2 (R)，Fourier变换$\hat{\phi}(w)$满足允许条件

$C_\phi = \int_R \frac{|\hat{\phi}(w)|^2}{|w|} \mathrm{d}w < \infty$，称φ(t)为一个基本小波或母小波。可以证明，小波一定满足这样的性质。

$$\int_{-\infty}^{\infty} \phi(t) \mathrm{d}t = 0$$

这也是我们称之为“小波”的意义。

将母函数经过伸缩和平移后，就可以得到小波序列为

$$\phi_{a,b}(t) = \frac{1}{\sqrt{|a|}} \phi(\frac{t-b}{a})(a, b \in R, a \neq 0) \tag{7-4}$$

任意的函数 f (t)∈L^2 (R)的连续小波变换为

$$W_f(a,b) = \langle f, \phi_{a,b} \rangle = \frac{1}{\sqrt{|a|}} \int_R f(t) \overline{\phi(\frac{t-b}{a})} \mathrm{d}t \tag{7-5}$$

记$\phi_a(t) = \phi(\frac{t}{a})$,可以看出，连续小波变换就是信号与小波函数的卷积。小波函数事实上就是我们信号处理的一个滤波器。其逆变换为

$$f(t) = \frac{1}{C_\phi} \int_{R^+} \int_R \frac{1}{a^2} W_f(a,b) \phi(\frac{t-b}{a}) \mathrm{d}a \mathrm{d}b \tag{7-6}$$

小波变换的实质在于将$L^2(R)$空间中的任意函数$f(t)$表示成为其在具有不同伸缩因子和平移因子上得投影的叠加。与傅立叶变换（仅将$f(t)$投影到频率域）不同的是，小波变换将一维时域函数映射到二维“时间-尺度”域上，因此$f(t)$在小波基上得展开具有多分辨率的特性。通过调整伸缩因子和平移因子，可以得到具有不同时频宽度的小波以匹配原始信号的任意位置，达到对信号的时频局部化分析的目的。

我们不妨做个粗略的比喻来解释小波变换的作用。我们用镜头观察目标$f(t)$，$\varphi(t)$代表镜头所起的作用（如滤波或卷积）。b相当于使镜头相对于目标平行移动，t的作用相当于镜头向目标推进或远离。由此可见，b仅仅影响时频窗口在相平面时间轴上的位置，而t不仅影响时频窗口在频率轴上的位置，也影响窗口的形状。这样小波变换对不同的频率在时域上得取样步长是可调节的，也就是多分辨率。

在式（7-5）中，对参数a、b进行展开后，就得到了任何时刻任意精度的频谱，但是对实际计算来讲，计算量太大，所以考虑将其离散化。

离散小波函数表示为

$$\phi_{j,k}(t) = a_0\phi(a_0^{-j}t - kb_0) \qquad (7\text{-}7)$$

离散小波系数表示为

$$W_{j,k}(t) = \int_{-\infty}^{\infty} f(t)\overline{\phi_{j,k}(t)}\mathrm{d}t \qquad (7\text{-}8)$$

选取$a_0=\frac{1}{2}$,$b_0=1$，则称为二进小波。

从理论上可以证明，将连续小波变换离散成离散小波变换，信号的基本信息并不会丢失。相反，由于小波基函数的正交性，使得小波空间中两点之间因冗余度造成的关联得以消除；同时，由于正交性，使得计算的误差更小，变换结果时-频函数更能反映信号本身的性质。

为了更好地理解小波变换，我们将其与第6章的傅立叶变换对比，可以得出以下结论。

（1）傅立叶变换是把能量有限的信号f(t)分解到以$\{e^{jwt}\}$为正交基的空间上去，小波变换的实质是把该信号分解到W_j所构成的空间上去。

（2）傅立叶变换用到的基本函数只有sin(wt)，而小波函数具有不唯一性。

（3）在频域中，傅立叶变换具有较好的局部化能力，特别是对于频率成分比较简单的确定性信号，傅立叶变换很容易把信号变为各频率成分的叠加和的形式；但在时域中，傅里叶变换没有局部化的能力。

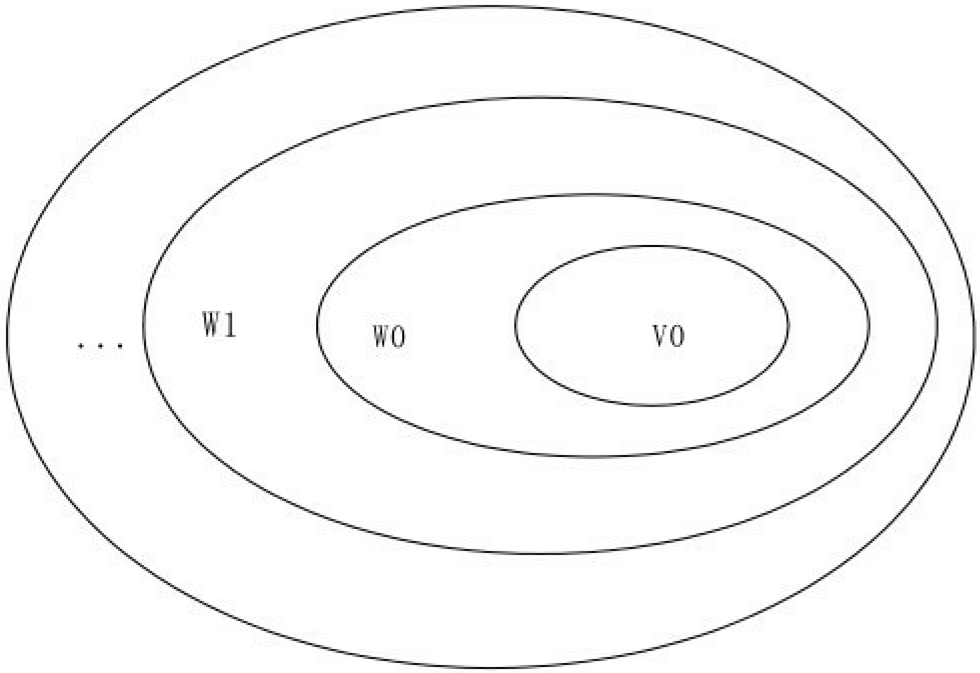
（4）小波分析中，尺度a的值越小相当于傅立叶变换中w的值越小。

多分辨率分析理论是建立在函数空间概念上的。数学中，函数空间是从集合X到集合Y的给定种类的函数的集合。将平方可积的函数f(t)∈L^2 (R)看成是某一逐级逼近的极限情况，每级逼近都是用某一低通平滑函数φ(t)对 f (t)做平滑的结果，在逐级逼近时平滑函数φ(t)也做逐级伸缩，这就是“多分辨率”，即用不同的分辨率来逐级逼近待分析函数 f (t)。

将空间做逐级二分解产生一组逐级包含的子空间。

$$\cdots,V_1=V_0\oplus W_0,V_2=V_1\oplus W_1,\cdots V_{j+1}=V_j\oplus W_j,\cdots \qquad (7\text{-}9)$$

j是从−∞到+∞的整数，j值越小空间越大，记号“⊕”表示子空间之和关系。当j=4时，如图7.4所示。



▲图7.4 函数空间的划分

由图7.4可见，空间剖分是完整的，即当$j\rightarrow-\infty$时，$v_j\rightarrow L^2(R)$,包含整个平方可积的实变函数空间。

$$\bigcup_{j\in z} V_j = L^2(R) \qquad (7\text{-}10)$$

当$j\rightarrow+\infty$时，$v_j\rightarrow 0$,即空间最终剖分到空集为止。

$$\bigcap_{j\in Z} V_j = \{0\} \qquad (7\text{-}11)$$

这种剖分方式使得空间v_j与空间W_j正交，各个W_j之间也正交，即

$$V_j \perp W_j, W_j \perp W_i, \qquad j \neq i \qquad (7\text{-}12)$$

综上可知，多分辨率分析即是指一串嵌套式子空间逼近序列$\{v_j\}_{j\varepsilon Z}$,它满足下列要求。

（1）一致单调性：对于任意的$j\in Z$,有$v_{j+1}\subset v_j$。

（2）逼近性：$\bigcap_{j\in Z} V_j=\{0\}$，$\bigcup_{j\in Z} V_j=L^2(R)$。

（3）伸缩性：$f(t)\in v_j \Leftrightarrow f(2^j t)\in v_0$,伸缩性体现了尺度的变化，逼近正交小波函数的变化和空间的变化具有一致性。

（4）平移不变性：对任意$k\in Z$,有$\varphi_j(2^{-j/2}t)\in v_j \Rightarrow \phi_j(2^{-j/2}t-k)\in v_j$（其中$\varphi(t)$称为尺度函数（后面详细介绍）。）

（5）正交基存在性：存在$\varphi(t)\in v_0$，使得$\{\phi_j(^-2^{j/2}t-k)|k\in Z\}$构成$v_j$的正交基。

如果对于满足多分辨率分析的一系列闭子空间$\{v_j\}_{j\in Z}$,$\phi_{j,k}(t)$是由$\{v_j\}_{j\in Z}$产生的尺度函数，用$\phi_{j,k}(t)$作小波，它的补空间为$\{W_j\}_{j\in Z}$,$v_j=v_{j+1}\oplus W_{j+1}$,则$\{W_j\}_{j\in Z}$构成小波空间。任意信号$f(t)\in L^2$（R）可用多分辨率分析公式表示为

$$f(t)=\sum_k c_{j,k}\phi_{j,k}(t)+\sum_j\sum_k d_{j,k}\varphi_{j,k}(t) \tag{7-13}$$

式中： j是任意开始尺度；$c_{j,k}$通常称为近似值或尺度系数；$D_{j,k}$称为细节或小波系数。这是因为式（7-13）的第一个和式用尺度函数提供了$f(x)$在尺度j的近似（除非$f(x)\in v_j$，此时为精确值）。对于第二个和式中每个较高尺度的j，更细分辨率的函数被添加到近似中，以获得细节的增加。如果展开函数形成了一个正交基或紧框架，则展开系数计算如下。

$$c_{j,k}=\left\langle f(x),\phi_{j,k}(x)\right\rangle=\int f(x)\phi_{j,k}(x)\mathrm{d}x \tag{7-14}$$

$$d_{j,k}=\left\langle f(x),\varphi_{j,k}(x)\right\rangle=\int f(x)\varphi_{j,k}(x)\mathrm{d}x \tag{7-15}$$

如果展开函数是双正交基的一部分，上式中的φ和ϕ项要分别由它们

的对偶函数和代替。

在上述多分辨率分析的概念介绍中，我们提到了尺度函数和小波函数的概念。怎样理解这两者的关系，我们来举个例子。事实上，可以假设在三维空间里表达一个向量，则需要建立一个三维的坐标系，只要坐标系建立，我们就可以用3个点（x,y,z）来简单地表示一个向量，同样，在一个信号我们设为f(t)，要想表示它，我们可以用一个个正交的简单函数来构建坐标系，然后将f(t)映射在这些简单的正交函数上，产生一个系数，这些系数我们就可以等同于(x,y,z)，也就是说利用相互正交的简单函数，构建一个表达信号的空间“坐标系”，然后就可以用这些系数和正交函数来表示f(t)。

在小波分析中，这个构建坐标系的函数就是小波函数，但是在小波函数来表示一个信号的时候，它其实是将信号映射在了时频平面内的，所以在实现过程中需要一个频域的底座和平台，来让信号f(t)与之做映射后是在一定的频率分辨率上进行的，这个起到底座的函数就是尺度函数，在尺度函数的平台下对频率的分析，或者说对信号的 f(t)的表达就是小波函数的作用了。在滤波实现中，低频滤波就相当于尺度函数的作用，小波函数的实现就是高频滤波器的使用。

[例 7.1] $y = x^2$ 的哈尔小波序列展开。

考虑图7.5（a）所示简单函数

$$y=\begin{cases} x^2 & 0\leqslant x<1 \\ 0 & \text{其他} \end{cases}$$

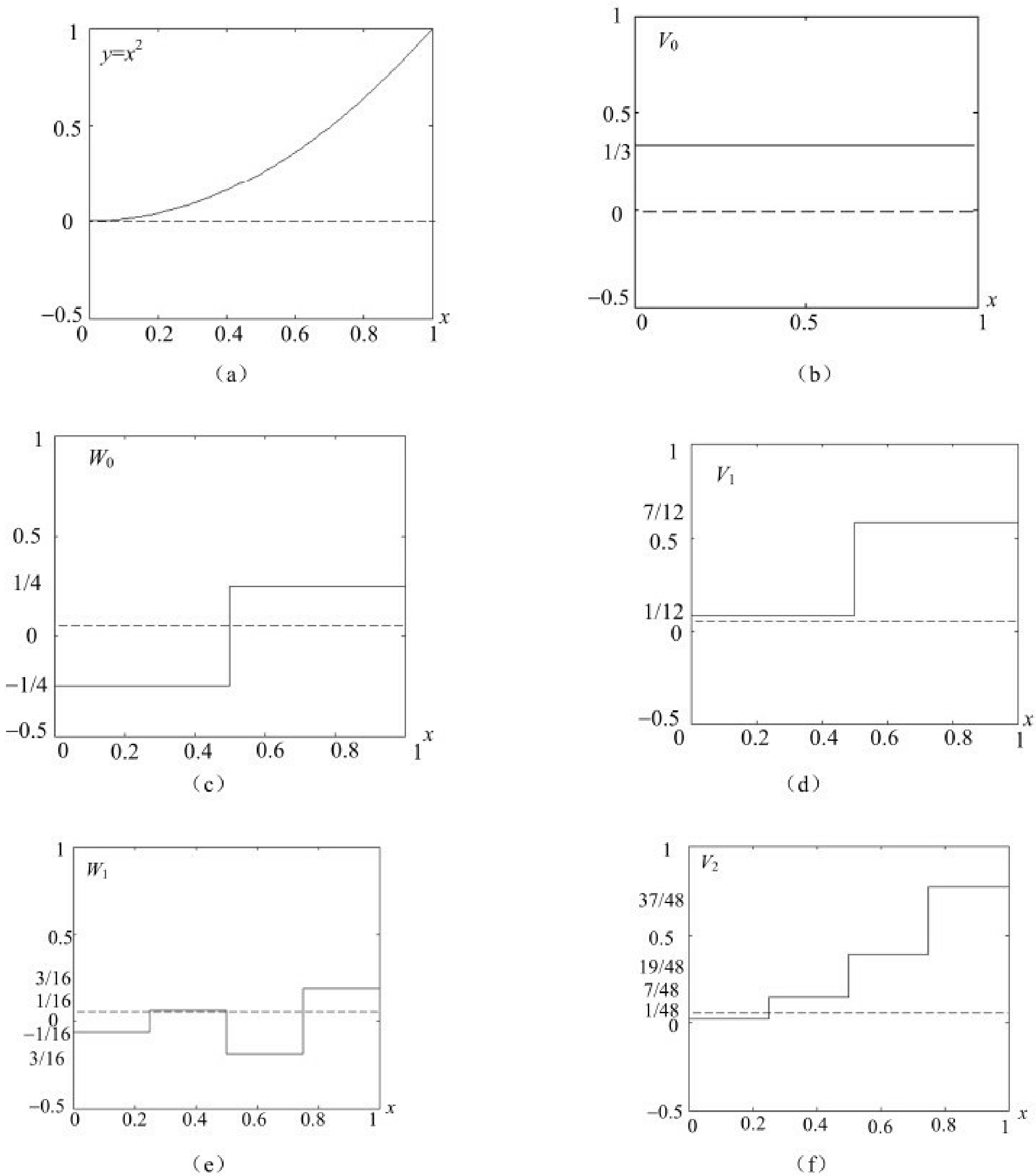
使用哈尔尺度函数 $$\phi(x)=\begin{cases} 1 & 0\leqslant x<1 \\ 0 & \text{其他} \end{cases}$$

哈尔小波函数如下

$$\varphi(x)=\begin{cases}1 & 0\leqslant x<0.5\\ -1 & 0.5\leqslant x<1\\ 0 & \text{其他}\end{cases}$$

根据式（7-6）和式（7-7）计算下列展开系数。

$$c_0(0)=\int_0^1 x^2\phi_{0,0}(x)\mathrm{d}x=\int_0^1 x^2\mathrm{d}x=\frac{1}{3}$$



▲图7.5 使用哈尔小波将y=x2的小波序列展开

$$d_0(0)=\int_0^1 x^2\varphi_{0,0}(x)\mathrm{d}x=\int_0^{0.5}x^2\mathrm{d}x-\int_{0.5}^1 x^2\mathrm{d}x=-\frac{1}{4}$$

$$d_1(0)=\int_0^1 x^2\varphi_{1,0}(x)\mathrm{d}x=\int_0^{0.25}x^2\sqrt{2}\mathrm{d}x-\int_{0.25}^{0.5}x^2\sqrt{2}\mathrm{d}x=-\frac{\sqrt{2}}{32}$$

$$d_1(1)=\int_0^1 x^2\varphi_{1,1}(x)\mathrm{d}x=\int_0^{0.75}x^2\sqrt{2}\mathrm{d}x-\int_{0.75}^{0.5}x^2\sqrt{2}\mathrm{d}x=-\frac{3\sqrt{2}}{32}$$

将这些值带入式（7-5），可以得到如下小波序列展开。

$$y=\underbrace{\underbrace{\underbrace{\frac{1}{3}\phi_{0,0}(x)}_{V_0}+\underbrace{[-\frac{1}{4}\varphi_{0,0}(x)]}_{W_0}}_{V_1=V_0\oplus W_0}+\underbrace{[-\frac{\sqrt{2}}{32}\varphi_{1,0}(x)-\frac{3\sqrt{2}}{32}\varphi_{1,1}(x)]}_{W_1}}_{V_2=V_1\oplus W_1=V_0\oplus W_0\oplus W_1}+\cdots$$

上述展开中的第一项用$c_0(0)$生成待展开函数的v_0子空间近似值。该近似值是原始函数的平均值。第二项使用$D_0(0)$通过从W_0子空间添加一级细节来修饰该近似值。其他级别的细节由子空间W_1的系数$D_1(0)$和$D_1(1)$给出。此时，展开函数现在已经接近原始函数了。越高的尺度（或细节的级别越高）被叠加，近似值越变得接近函数的精确表示，它的极限是$j\rightarrow\infty$。

7.1.2 分解与重构的实现

根据7.1.1节的论述，我们可以知道，在理解多分辨率分析时，需要把握一个要点：分解的最终目的是力求构造一个在频率上高度逼近的$L^2(R)$空间的正交小波基，这些频率分辨率不同的正交小波基相当于带宽各异的带通滤波器。另外，多分辨率分析只对低频空间作进一步分

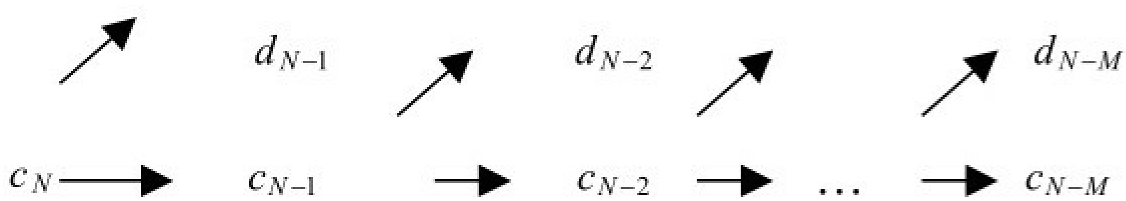
解，使频率的分辨率越来越高。

根据7.1.1节中的式（7-5），其中的剩余系数$c_{j,k}$和小波系数$D_{j,k}$,利用分解重构公式，得

$$
\begin{aligned}
c_{j,k} &= \sum_m h_0(m-2k)c_{j-1,m} \\
d_{j,k} &= \sum_m h_1(m-2k)c_{j-1,m}
\end{aligned}
\qquad (7\text{-}16)
$$

式（7-5）中等号右边第一部分是 f (t)在尺度空间v_j的投影，是 f (t)的平滑近似；第二部分是f (t)在小波空间W_j的投影，是对 f (t)的细节补充。式（7-16）是小波分解系数的递推计算公式，式中$H_0(k)$和$H_1(k)$分别为低通和高通数字滤波器单位取样响应。

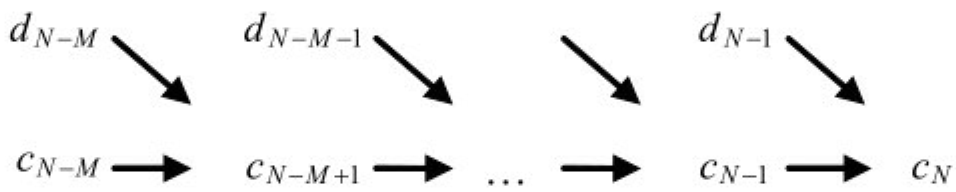
于是，若要将信号由N水平分解到N-M水平，则可得分解过程为



对于任意函数 f (t)∈v_{j-1}，将其分解一次投影到v_j、W_j空间，即可分别得到剩余系数$c_{j,k}$和小波系数$D_{j,k}$重建原始信号，下式是变换系数的重建公式。

$$
c_{j-1,m} = \sum_k c_{j,k} h_0(m-2k) + \sum_k d_{j-k} h_1(m-2k) \qquad （7\text{-}17）
$$

形象表示此重构过程为

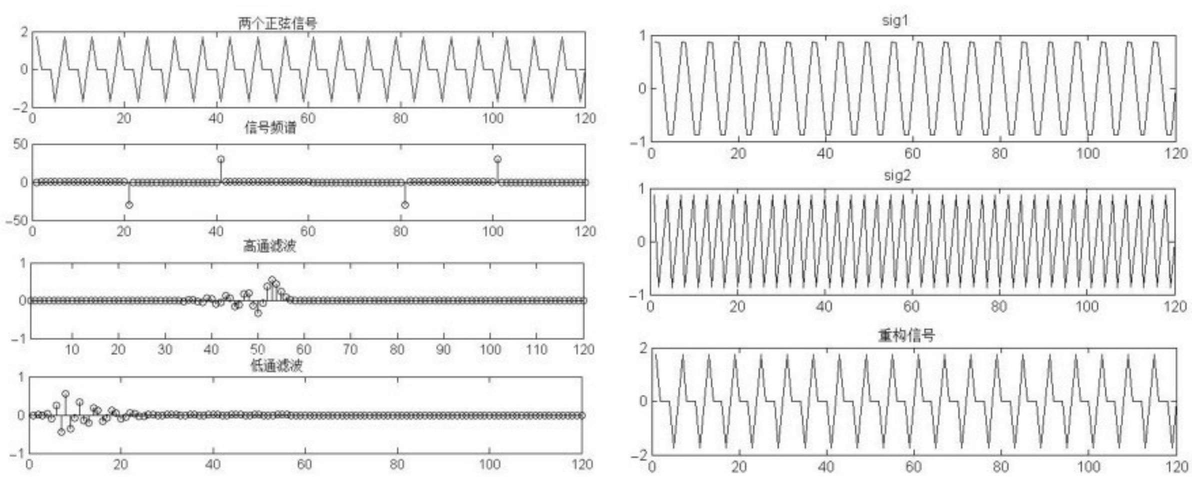


[例7.2] 信号分解与重构的Matlab实现，通过对由两个正弦信号叠加的信号进行分解重构，我们可以了解基本信号分解与重构的过程怎样用Matlab实现。

```
%信号函数定义
% 1.正弦波定义
f1=50;    % 频率1
f2=100;    % 频率2
fs=2*(f1+f2);    % 采样频率
Ts=1/fs;    % 采样间隔
N=120;    % 采样点数
n=1:N;
●=sin(2*pi*f1*n*Ts)+sin(2*pi*f2*n*Ts); % 信号函数
% 2小波滤波器谱分析
h=wfilters('db30','l');    % 低通
g=wfilters('db30','h');    % 高通
h=[h,zeros(1,N-length(h))];    % 补零（圆周卷积，且增大分辨率便于观察）
g=[g,zeros(1,N-length(g))];    % 补零（圆周卷积，且增大分辨率便于观察）
% 3 MALLAT分解算法(圆周卷积的快速傅立叶变换实现)
sig1=ifft(fft(y).*fft(h));    % 低通(低频分量)
sig2=ifft(fft(y).*fft(g));    % 高通(高频分量)
%4 MALLAT重构算法
```

```
sig1=dyaddown(sig1);    % 2抽取
sig2=dyaddown(sig2);    % 2抽取
sig1=dyadup(sig1);    % 2插值
sig2=dyadup(sig2);    % 2插值
sig1=sig1(1,[1:N]);    % 去掉最后一个零
sig2=sig2(1,[1:N]);    % 去掉最后一个零
hr=h(end:-1:1);    % 重构低通
gr=g(end:-1:1);    % 重构高通
hr=circshift(hr',1)';    % 位置调整圆周右移一位
gr=circshift(gr',1)';    % 位置调整圆周右移一位
sig1=ifft(fft(hr).*fft(sig1));    % 低频
sig2=ifft(fft(gr).*fft(sig2));    % 高频
sig=sig1+sig2;    % 源信号
```

程序运行结果如图7.6所示。从图7.6中可以看出原始信号即是两个正弦信号的叠加，在经过高通和低通分解后，得到两个高频和低频的信号 sig1 与 sig2；再经过重构后，两部分又重构为原始信号。



▲图7.6 程序运行结果图

7.1.3 图像处理中分解与重构的实现

前面讲述了一维信号的多分辨率分解与合成算法，对于二维的图像信号，我们可以从滤波器的角度来理解多分辨率分析。首先对图像先“逐行”作一维小波变换，分解为低通滤波L和高通滤波H两个分量，再“逐列”作一维小波变换，分解为LL、LH、HL、HH 4个分量。L和H分别表示低通和高通滤波输出。

相应的，二维尺度函数φ(x, y)=φ(x)φ(y)。

二维小波函数有3个，对应不同方向上的高/低通滤波特性。

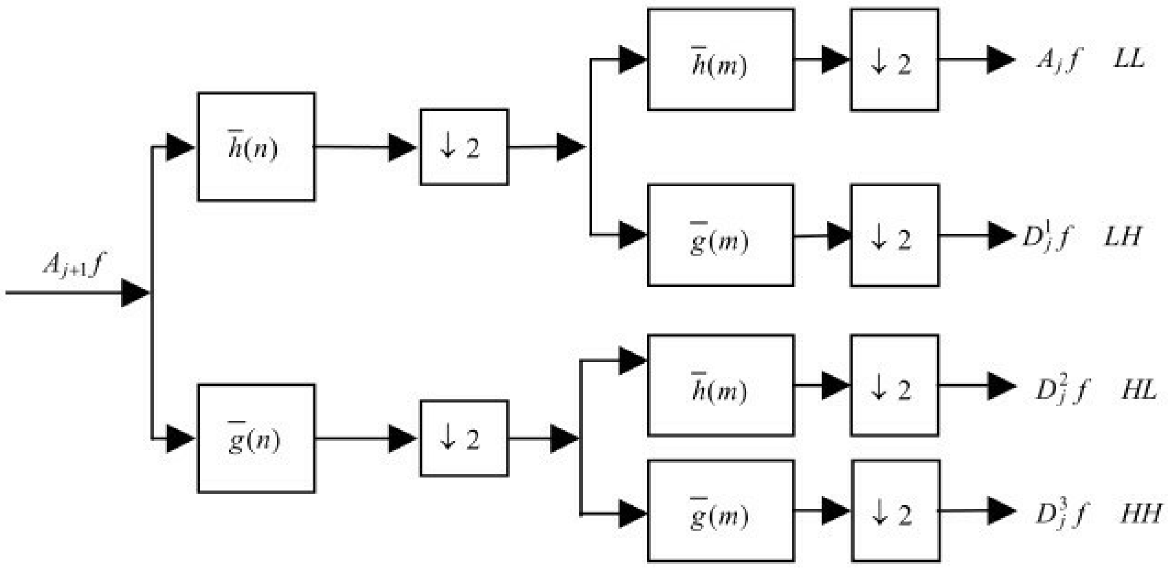
$$\begin{cases} \varphi^1(x,y) = \varphi(x)\varphi(y), LH \\ \varphi^2(x,y) = \varphi(x)\varphi(y), HL \\ \varphi^3(x,y) = \varphi(x)\varphi(y), HH \end{cases}$$

分解的结果，在2^j 层次有$A_j f$（逼近）及$D_j^1 f$,$D_j^2 f$,$D_j^3 f$ 3个细节信号。所以每“上”一层，近似图像分解为4个分量。若原图像为A_0,分解总层数为J，则共有3J+1幅子图像。分解合成过程可以表示为

分解：$A_{j+1}f \rightarrow (A_j f,(D_j^1 f, D_j^2 f, D_j^3 f))$

合成：$A_{j+1}f = A_j f + D_j^1 f + D_j^2 f + D_j^3 f$

下图给出图像信号的快速小波分解图。



▲图7.7 图像的快速小波分解框图

[例7.3] 图像处理中小波变换的应用

图7.4（a）所示为测试图像，通过这幅图片我们可以看出二维小波变换的方向敏感性和边缘检测的有效性。程序中利用小波工具箱中的函数，简便快捷。

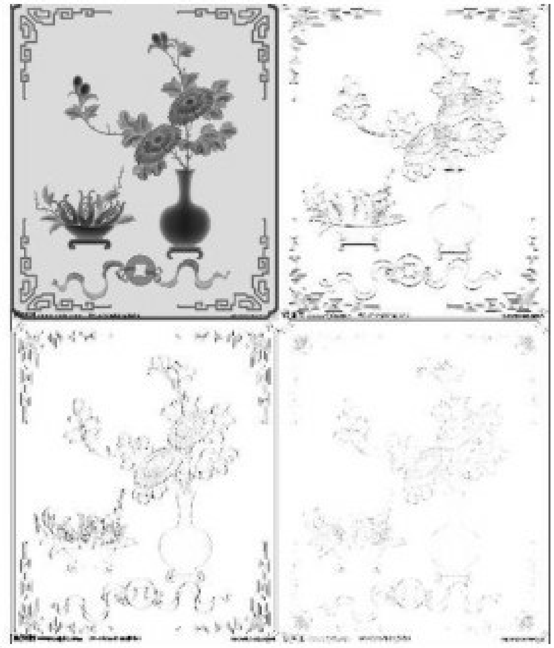
```
f=imread('D:\1.jpg');    %读入一幅图
imshow(f);    %显示读入的灰度图
[c,s]=wavefast(f,1,'sym4');    %利用'sym4'小波作快速小波变换
figure;wave2gray(c,s,-6);    %利用小波工具箱的函数显示变换后灰度图
[nc,y]=wavecut('a',c,s);    %利用小波工具箱的函数将近似系数置0
figure; wave2gray(nc,s,-6);
edges=abs(waveback(nc,s,'sym4'));    %边缘图像的重构
figure; imshow(mat2gray(edges));
```

图7.4（a）所示的单尺度小波变换的水平、垂直、对角线的方向性在图7.4（b）中可清楚地看到。注意，原图像的水平边缘出现在图

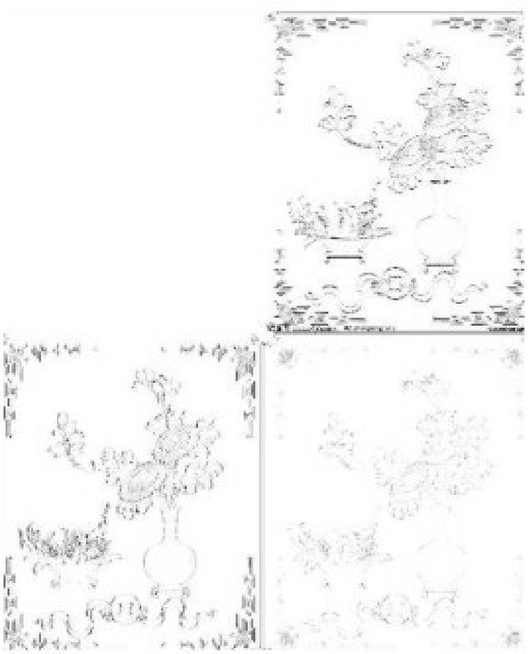
7.4（b）所示的右上象限的水平细节系数中。对于图像的垂直边缘，可以在左下象限的垂直细节系数中类似地确定。要将这些信息合并成一幅边缘图像，我们可以简单地把生成的变换的近似系数设为零，计算它的反变换，再对其取绝对值。修改过的变换和得到的边缘图像分别显示在图7.4（c）和图7.4（d）中。类似的过程可用于隔离垂直边缘或水平边缘。



（a）测试图



（b）图像的小波变换的分解



（c）将所有近似系数设置为 0 的变换



（d）重构得到的边缘图像

▲图7.8 图像处理中边缘检测中的小波变换

[例7.4] 利用小波分析对图像去噪

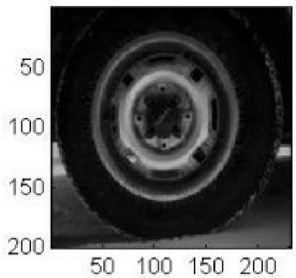
装入一幅图像，并对其添加噪声，利用小波对添加噪声后的图像进

行小波分解，并观察滤波效果。

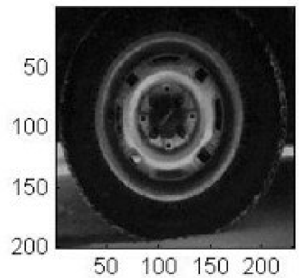
程序如下。

```
%装入图像
load tire
%下面进行噪声的产生
init=3718025452;
rand('seed',init);
Xnoise=X+18*(rand(size(X)));
%显示原始图像及它的含噪声的图像
colormap(map);
subplot(2,2,1);image(wcodemat(X,192));
title('原始图像')
axis square
subplot(2,2,2);image(wcodemat(Xnoise,192));
title('含噪声的图像');
axis square
%用sym5小波对图像信号进行二层的小波分解
[c,s]=wavedec2(X,2,'sym5');
%下面进行图像的去噪处理
%使用ddencmp函数来计算去噪的默认阈值和熵标准
%使用wdencmp函数来实现图像的压缩
[thr,sorh,keepapp]=ddencmp('den','wv',Xnoise);
[Xdenoise,cxc,lxc,perf0,perfl2]=wdencmp('gbl',c,s,'sym5',2,thr,sorh,keep
%显示去噪后的图像
subplot(2,2,3);image(Xdenoise);
title('去噪后的图像');
axis square
```

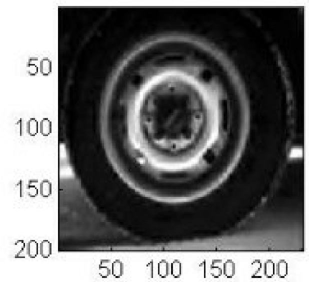
运行结果如图7.9所示，对比原始图像和去噪后的图像，利用小波去噪后，图像比原图像更亮一些。



（a）原始图像



（b）含噪声的图像



（c）去噪后的图像

▲图7.9 小波分析用于图像去噪

[例7.5] 小波变换用于图像压缩

装入一幅图像，使用哈尔小波对图像进行三层分解，利用函数wdcbm获取压缩阈值，函数wdencmp用于对信号进行压缩。将图像分解为低频信息和高频信息，图像的主要成分包含在低频信息中，对低频信息进行两次小波分解，并改变图像的大小进行压缩，分别获取两次压缩后的图像。

程序如下。

```
%装载信号
load nelec;
indx=1:1024;
x=nelec(indx);
% 用小波haar对信号进行三层分解
[c,l]=wavedec(x,3,'haar');
alpha=1.5;
% 获取信号压缩的阈值
[thr,nkeep]=wdcbm(c,l,alpha);
% 对信号进行压缩
```

```
[xd,cxd,lxd,perf0,perfl2]=wdencmp('lvd',c,l,'haar',3,thr,'s');
subplot(2,1,1);
plot(indx,x);
title('原始信号');
subplot(2,1,2);
plot(indx,xd);
title('压缩后的信号');
>> %装入图像
load tire
%下面进行噪声的产生
init=3718025452;
rand('seed',init);
Xnoise=X+18*(rand(size(X)));
%显示原始图像及它的含噪声的图像
colormap(map);
subplot(2,2,1);image(wcodemat(X,192));
title('原始图像')
axis square
subplot(2,2,2);image(wcodemat(Xnoise,192));
title('含噪声的图像');
axis square
%用sym5小波对图像信号进行二层的小波分解
[c,s]=wavedec2(X,2,'sym5');
%下面进行图像的去噪处理
%使用ddencmp函数来计算去噪的默认阈值和熵标准
%使用wdencmp函数来实现图像的压缩
[thr,sorh,keepapp]=ddencmp('den','wv',Xnoise);
```

```
[Xdenoise,cxc,lxc,perf0,perfl2]=wdencmp('gbl',c,s,'sym5',2,thr,sorh,keep
%显示去噪后的图像
subplot(2,2,3);image(Xdenoise);
title('去噪后的图像');
axis square
>> %装入图像
load wbarb;
%显示图像
subplot(2,2,1);image(X);colormap(map)
title('原始图像');
axis square
disp('压缩前图像X的大小：');
whos('X')
%对图像用bior3.7小波进行2层小波分解
[c,s]=wavedec2(X,2,'bior3.7');
%提取小波分解结构中第一层低频系数和高频系数
ca1=appcoef2(c,s,'bior3.7',1);
ch1=detcoef2('h',c,s,1);
cv1=detcoef2('v',c,s,1);
cd1=detcoef2('d',c,s,1);
%分别对各频率成分进行重构
a1=wrcoef2('a',c,s,'bior3.7',1);
h1=wrcoef2('h',c,s,'bior3.7',1);
v1=wrcoef2('v',c,s,'bior3.7',1);
d1=wrcoef2('d',c,s,'bior3.7',1);
c1=[a1,h1;v1,d1];
%显示分解后各频率成分的信息
```

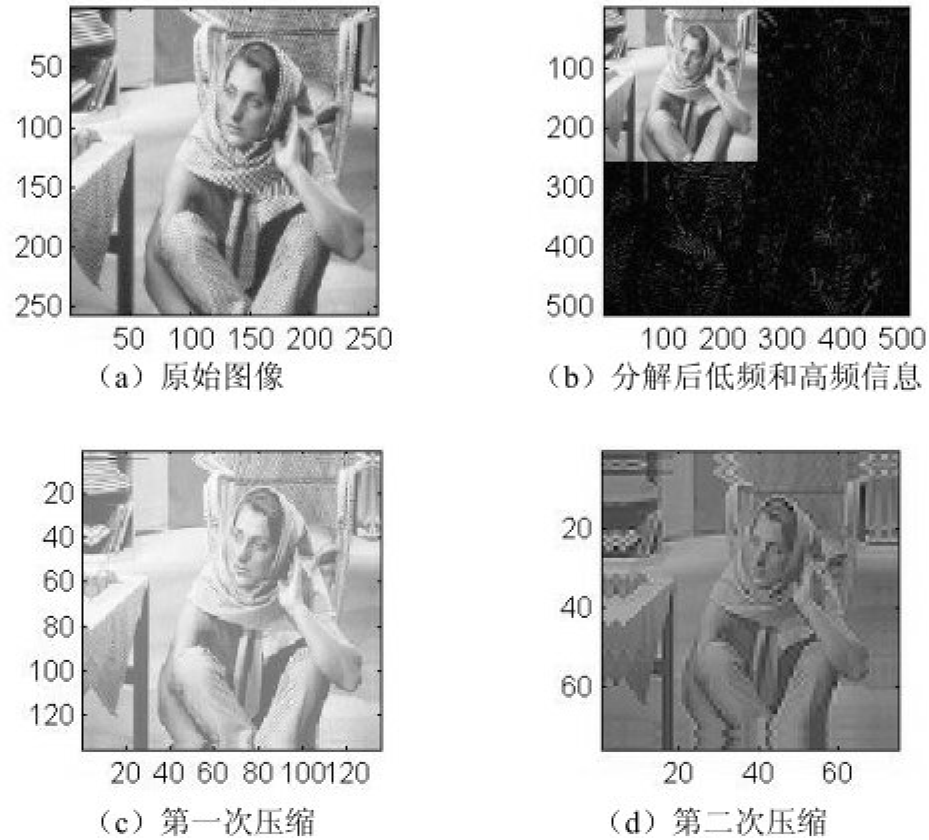
```
subplot(2'2'2);image(c1);
axis square
title('分解后低频和高频信息');
%下面进行图像压缩处理
%保留小波分解第一层低频信息，进行图像的压缩
%第一层的低频信息即为ca1，显示第一层的低频信息
%首先对第一层信息进行量化编码
ca1=appcoef2(c,s,'bior3.7',1);
ca1=wcodemat(ca1,440,'mat',0);
%改变图像的高度
ca1=0.5*ca1;
subplot(2,2,3);image(ca1);colormap(map);
axis square
title('第一次压缩');
disp('第一次压缩图像的大小为：');
whos('ca1')
%保留小波分解第二层低频信息，进行图像的压缩，此时压缩比更大
%第二层的低频信息即为ca2，显示第二层的低频信息
ca2=appcoef2(c,s,'bior3.7',2);
%首先对第二层信息进行量化编码
ca2=wcodemat(ca2,440,'mat',0);
%改变图像的高度
ca2=0.25*ca2;
subplot(2,2,4);image(ca2);colormap(map);
axis square
title('第二次压缩');
```

```
disp('第二次压缩图像的大小为：');
whos('ca2')
```

输出结果如下所示。

```
压缩前图像X的大小：
  Name        Size         Bytes  Class      Attributes
  X         256x256      524288 double
第一次压缩图像的大小为：
  Name        Size         Bytes  Class      Attributes
  ca1        135x135      145800 double
第二次压缩图像的大小为：
  Name        Size         Bytes  Class      Attributes
  ca2        75x75        45000  double
```

运行结果如图 7.10 所示，图中显示了经过分解后的低频信息和高频信息成分，再对低频信息成分进行压缩，将低频信息进行二次分解，分解后再进行压缩，也就是图 7.10 中所示的第一次压缩和第二次压缩的情况。可以看出，第一次压缩提取的是原始图像中小波分解第一层的低频信息，此时压缩效果较好，压缩比较小；第二次压缩是提取第一层分解低频部分的低频部分（即小波分解第二层的低频部分），其压缩比较大，压缩效果在视觉上一般，这是一种最简单的压缩方法只保留原始图像中低频信息，不经过其他处理即可获得较好的压缩效果。在上面的例子中，我们还可以只提取小波分解第3、第4层的低频信息。从理论上说，可以获得任意压缩比的压缩图像。



▲图7.10 利用二维小波分析进行图像压缩

7.2 Gabor多分辨率分析

前面讲到，傅立叶变换能将信号的时域特征和频域特征联系起来，能分别从信号的时域和频域进行分析，但却不能把二者有机地结合起来，这是因为信号的时域波形中不包含任何频域信息，而频域波形中又不包含任何时域信息。也就是说，傅立叶变换是时域与频域完全分离的，对于傅立叶频谱中的某一频率，无法知道这个频率是什么时候产生的。傅立叶变换在时域和频域局部化的问题上就显示出了它的局限性。

早在 1946 年，Gabor注意到傅立叶变换在表示非平稳信号方面的不足，他通过与量子力学中的不确定性原理的类比，发现并证明了一维信号的不确定性原理，即一个同时用时间和频率来刻画的信号特征受它的

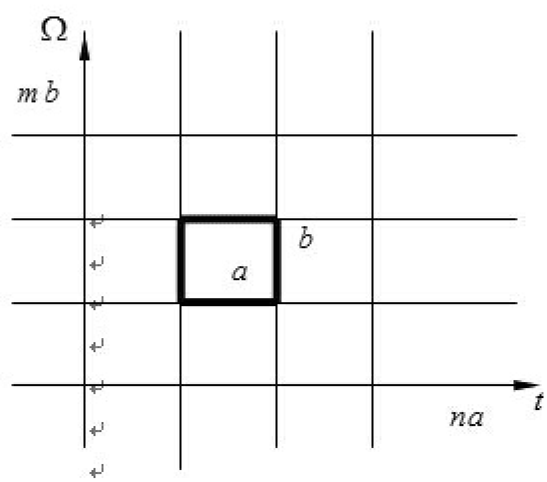
带宽和持续时间乘积的下限所限制。他提出了一种新的算法：Gabor变换。这种变换的基本思想是把信号划分成许多小的时间间隔，用傅立叶变换分析每一个时间间隔，以便确定信号在该时间间隔存在的频率。

我们从Gabor展开的基本概念来说明其原理。对于一个一维信号，我们可以用二维的时频平面上离散栅格处的点来表示一个一维的信号，即

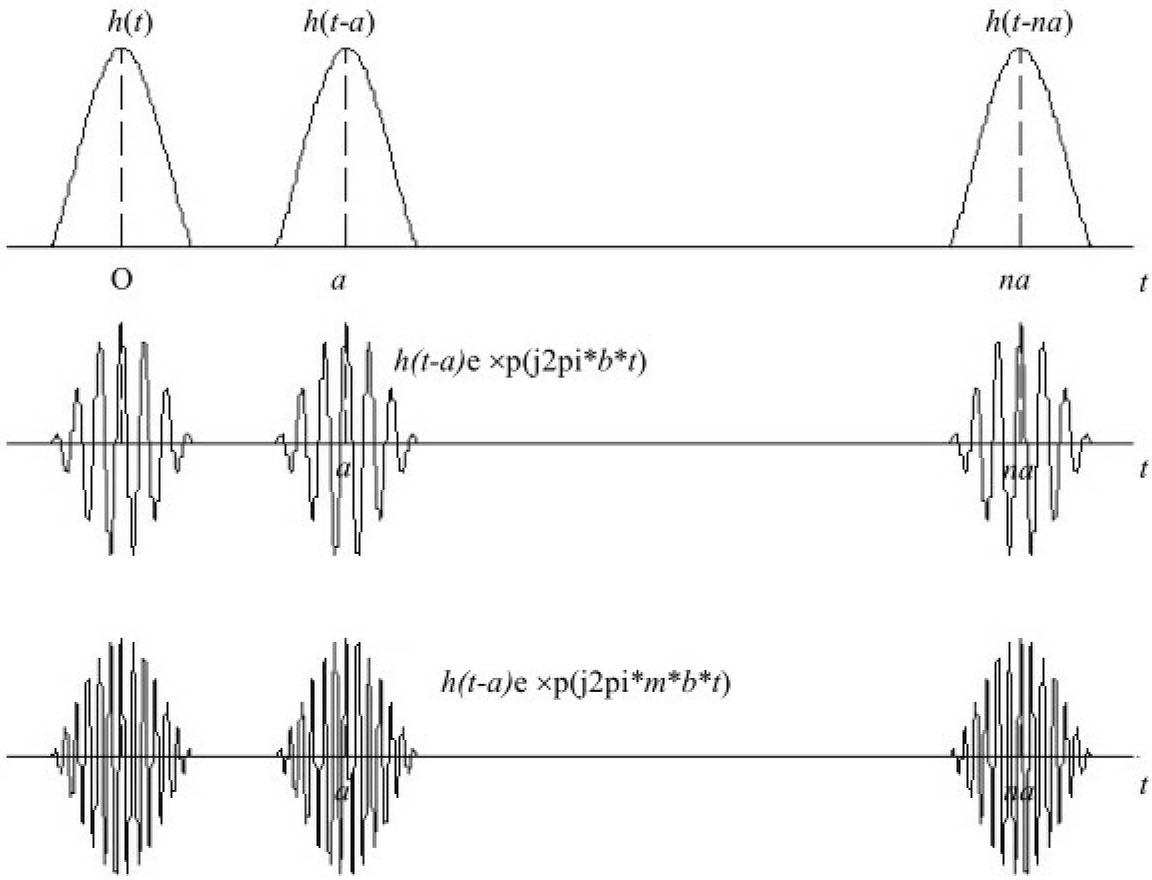
$$
\begin{aligned}
x(t) &= \sum_{m=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} C_{m,n} h_{m,n}(t) \\
&= \sum_{m=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} C_{m,n} h(t-na) \mathrm{e}^{\mathrm{j}2\pi mbt}
\end{aligned} \tag{7-18}
$$

式中：a,b为常数，a代表栅格的时间长度，b代表栅格的频率长度，如图7.11所示。

式（7-18）中的$c_{m,n}$是一维信号x(t)的展开系数，h(t)是一母函数，展开的基函数$H_{m,n}(t)$是由h(t)作移位和调制生成的如图7.12所示。



▲图7.11 Gabor展开的抽样栅格



▲图7.12 移位与调制后图像

Gabor最初选择高斯函数作为母函数h(t)，是因为高斯函数的傅立叶变换也是高斯的，因此保证了时域和频域的能量都相对较为集中。由于高斯信号的时宽-带宽积满足不定原理，因而保证了使用高斯信号可以得到最好的时间、频率分辨率。后续的研究表明，不止是高斯函数，其他的窗函数也都可以用来构成式（7-18）中的基函数。

可以证明，如果 ab>1，即栅格过稀，我们将缺乏足够的信息来恢复原信号 x(t)。当然，如果ab过小，必然会出现信息的冗余，这类似于对一维信号抽样时抽样频率过大的情况。因此，当ab=1时，称为临界抽样；ab>1时，称为欠抽样；ab<1时，称为过抽样。

要从时频二维空间分析信号，首先必须构造同时用时间和频率表示

的时间函数。Gabor在1946年提出了这样一种构造函数——复谱图。

$$F_{\phi,g}(t_0,w_0)=\left\langle \phi(t),g(t-t_0)\mathrm{e}^{\mathrm{j}w_0t}\right\rangle=\int_{-\infty}^{\infty}\phi(t)g^*(t-t_0)\mathrm{e}^{-\mathrm{j}w_0t}\mathrm{d}t \tag{7-19}$$

式中：φ(t)为信号；g(t)为窗函数； $\int_{-\infty}^{+\infty}|g(t)|^2\ \mathrm{d}t=1$ 。我们构造的是φ(t)与g(t)的时间平移g(t−t_0)和频率调制形式e^{-jw_0t}的复数共轭的内积。$f_{φ,g}(t_{0,}W_0)$其实是φ(t)g^* (t−t_0)的Fourier变换。利用求逆公式，可以从复谱图中重构原信号φ(t)。

$$\phi(t)=\frac{1}{2\pi}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}F_{\phi,g}(t_0,w_0)g(t-t_0)\mathrm{e}^{\mathrm{j}w_0t}\mathrm{d}t_0\mathrm{d}w_0 \tag{7-20}$$

重构信号φ(t)，并不需要知道全部复谱图，只需知道复谱图在一组网格点(t_0=mT,W_0=nΩ,m、n)为整数的值即可，其中ΩT =2π（临界采样）。

复谱图在网格点的取值 f_{mn}为

$$f_{mn}=\frac{1}{T}\int_{-\infty}^{\infty}\phi(t)g^*(t-mT)\mathrm{e}^{-\mathrm{j}n\Omega t}\mathrm{d}t \tag{7-21}$$

由于欠抽样时固有的缺点，因此人们很少研究它。Gabor最早提出的是使用高斯窗，并取临界抽样。但是，Gabor展开的这一想法长期没有被重视，其主要原因是由于展开系数计算的困难。直到1980年，Bastiaans提出了用建立辅助函数和对偶函数来求解展开系数的方法之后，Gabor展开的研究才引起了人们的兴趣。

从理论上讲，Gabor展开的讨论和时频分布、滤波器组及小波变换等新的信号处理理论密切相关。因此，这些新的信号处理理论的应用也涉及到Gabor展开的应用。Gabor展开在信号、图像的表示以及语音分析、目标识别、信号的瞬态检测等各方面都取得了较好的应用成果。

下面介绍一下连续信号Gabor展开系数的计算。

令

$$g_{m,n}(t)=g(t-na)\mathrm{e}^{\mathrm{j}2\pi mbt} \tag{7-22}$$

并令

$$C_{m,n}=\langle x(t),g_{m,n}(t)\rangle=\int x(t)g^{*}(t-na)\mathrm{e}^{-\mathrm{j}2\pi mbt}\mathrm{d}t \tag{7-23}$$

比较式（7-22）与式（7-23），立即发现

$$C_{m,n}(t)=STFT_{x}(m,n) \tag{7-24}$$

即Gabor系数是在离散栅格上求出的STFT。通常，式（7-23）称为Gabor变换，式（7-22）称为Gabor展开。

将式（7-24）代入式（7-22）中，有

$$\begin{aligned}x(t)&=\sum_{m}\sum_{n}\langle x(t),g_{m,n}(t)\rangle h_{m,n}(t)\\&=\sum_{m,}\sum_{n}[\int x(t')g_{m,n}^{*}(t')\mathrm{d}t']h_{m,n}(t)\\&=\int x(t')[\sum_{m}\sum_{n}g_{m,n}^{*}(t')h_{m,n}(t)]\mathrm{d}t'\end{aligned} \tag{7-25}$$

若要该式的右边等于x(t)，则必有

$$\sum_{m}\sum_{n}g_{m,n}(t')h_{m,n}(t)=\delta(t-t') \tag{7-26}$$

式（7-26）给出了为保证由$c_{m,n}$恢复x(t)、$H_{m,n}(t)$和$g_{m,n}(t)$应遵循的条件。满足该条件的$H_{m,n}(t)$被称为是完备的。

由式（7-26）还可以引申出母函数h(t)和其对偶母函数g(t)之间的关

系。

$$\int g(t)h^*(t-na)\mathrm{e}^{-\mathrm{j}2\pi mbt}\mathrm{d}t=\delta_m\delta_n \tag{7-27}$$

该式称为g(t)和h(t)之间的双正交关系。显然，若m,n中有一个不为零，上式的积分即为零。若m=n=0，则

$$\int g(t)h^*(t)\mathrm{d}t=1 \tag{7-28}$$

以上给出的关系是在 ab=1，即临界抽样的情况下得到的。由上面的讨论，我们可得到一个求解Gabor系数的方法。

（1）选择一个母函数（或基函数）h(t)。

（2）求其对偶函数g(t)，使之满足（7.2.10）式及（7.2.11）式。

（3）按（7.2.9）式做内积，从而得到$c_{m,n}$。

上面的分析表明，任意可以用高斯函数调制的复正弦形式表示的信号都可以达到时域和频域联合不确定关系的下限，可以同时在时域和频域获得最佳的分辨率，这种表示是Gabor函数的最初形式。

最近二三十年，随着神经生理学和小波变换技术的发展，Gabor函数逐渐演变为二维小波的形式。二维Gabor小波变换是图像的多尺度表示和分析的有力工具，作为唯一能够取得空域和频域联合不确定关系下限的Gabor函数经常被用作小波基函数，对图像进行各种分析。前一节的内容告诉我们小波变换是用一组滤波器函数与给定信号的卷积来表示或逼近一个信号。二维Gabor滤波器的函数形式可以表示为

$$\varphi_j(x)=\frac{\|\overrightarrow{k_j}\|^2}{\sigma^2}\exp\left(-\frac{\|\overrightarrow{k_j}\|^2\|\vec{x}\|^2}{2\sigma^2}\right)\left[\exp(i\overrightarrow{k_j}\vec{x})-\exp\left(-\frac{\sigma^2}{2}\right)\right] \tag{7-29}$$

$$\overrightarrow{k_j}=\binom{K_{jx}}{k_{jy}}=\binom{k_v\cos\varphi_u}{k_v\sin\varphi_u} \tag{7-30}$$

式中：$\vec{x}$为给定位置的图像坐标；$\vec{k_j}$为滤波器的中心频率；ϕ_u体现了滤波器的方向选择性。在自然图像中，$\frac{\|\vec{k_j}\|^2}{\sigma^2}$用来补偿由频率决定的能量谱衰减。$\exp(-\frac{\|\vec{k_j}\|^2\|\vec{x}\|^2}{2\sigma^2})$用来约束平面波的高斯包络函数。$\exp(i\vec{k_j}\vec{x})$为复数值平面波，其实部为余弦平面波$\cos(\vec{k_j}x)$，虚部为正弦平面波$\sin(\vec{k_j}x)$；由于余弦平面波关于高斯窗口中心偶对称，在高斯包络函数的约束范围内，其积分值不为0；而正弦平面波关于高斯窗口奇对称，在高斯包络函数的约束范围内，其积分值为0；为了消除图像的直流成分对二维Gabor小波变换的影响，在复数值平面波德实部减去$\exp(-\frac{\sigma^2}{2})$，这使得二维Gabor小波变换不受图像灰度绝对数值的影响，并且对图像的光照变化不敏感。

二维Gabor滤波器是带通滤波器，在空间域和频率域均有较好的分辨能力，它在空间域有良好的方向选择性，在频率域有良好的频率选择性，二维Gabor小波可以提取图像不同的频率尺度和纹理方向的信息。二维Gabor小波滤波器组的参数体现了它在空间域和频率域的采样方式，决定了它对信号的表达能力。

二维Gabor小波是由二维Gabor滤波器函数通过尺度伸缩和旋转生成的一组滤波器，其参数的选择通常在频率空间进行考虑。为了对一幅图像的整个频率进行采样，可以采用具有多个中心频率和方向的Gabor滤波器组来描述图像。参数k_φ、ϕ_u的不同选择分别体现了二维Gabor小波在频率和方向空间的采样方式。参数σ决定滤波器的带宽，两者的关系为

$$\sigma=\sqrt{2\ln 2}\left(\frac{2^{\phi}+1}{2^{\phi}-1}\right) \qquad （7-31）$$

式中：φ为用倍频程表示的半峰带宽，当φ=0.5倍频程时，σ≈2π；当φ=1倍频程时，σ≈π；当φ=1.5倍频程时，σ≈2.5。

7.3 常见小波分析

小波变换的基本思想是用一组小波函数或者基函数表示一个函数或者信号。信号分析一般是为了获得时间和频率域之间的相互关系，傅立叶变换提供了有关频率域的信息，但时间方面的局部化信息却基本丢失。与傅立叶变换不同，小波变换通过平移母小波或基本小波可获得信号的时间信息，而通过缩放小波的宽度或尺度可获得信号的频率特性。在小波变换中，近似值是大的缩放因子产生的系数，表示信号的低频分量。而细节值是小的缩放因子产生的系数，表示信号的高频分量。

与标准的傅立叶变换相比，小波分析中所用到的小波函数具有不唯一性，即小波函数具有多样性。小波分析在应用中的一个十分重要的问题就是最优小波基的选择问题，因为用不同的小波基分析同一个问题会产生不同的结果。目前我们主要是通过用小波分析方法处理信号的结果与理论结果的误差来判定小波基的好坏，由此决定小波基函数。

虽然根据不同的标准，小波函数具有不同的类型，这些标准通常是下面几点。

(1)ϕ(t)、ϕ(w)、φ(t)和φ(w)的支撑长度，即当时间或频率趋向无穷大时，ϕ(t)、ϕ(w)、φ(t)和φ(w)从一个有限值收敛到0（注：φ(t)为尺度函数，小波函数ϕ(t)可以由它求出来）。

（2）对称性，它在图像处理中可以很有效地避免移相。

（3）正则性，它在对信号或图像的重构获得较好的平滑效果作用

上是非常有用的。

具有对称性的小波不产生相位畸变；具有好的正则性的小波易于获得光滑的重构曲线和图像，从而可以减小误差。

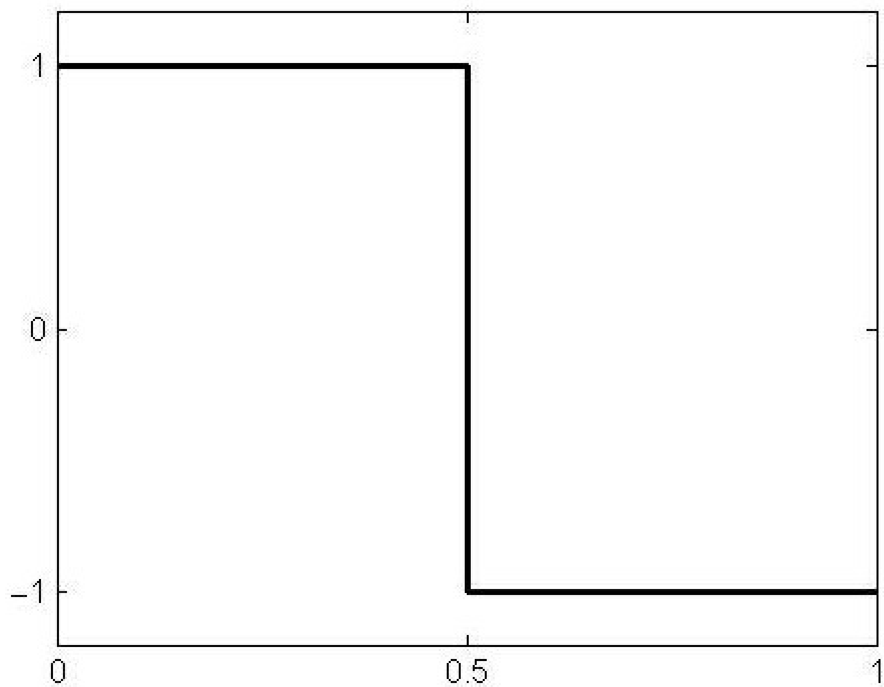
在本节中，我们主要介绍常用的Haar小波和Daubechies小波。

7.3.1 Haar小波

Haar 函数是小波分析中最早用到的一个具有紧支撑地正交小波函数，也是最简单的一个小波函数，它是支撑域在t∈[0,1]范围内的单个矩形波。Haar函数的定义如下。

$$\varphi(t)=\begin{cases}1 & 0\leqslant t\leqslant \frac{1}{2}\\ -1 & \frac{1}{2}\leqslant t\leqslant 1\\ 0 & \text{其他}\end{cases} \qquad （7\text{-}32）$$

Haar小波的形状图如图7.13所示。



▲图7.13 Haar小波形状图

Haar 小波在时域上是不连续的，所以作为基本小波性能不是特别好。但它也有自己的优点，具体如下。

（1）计算简单。

（2）$\phi(t)$不但与$\phi(2^j t)[j\in Z]$正交（$\int\phi(t)\phi(2^j t)dt=0$），而且与自己的整数位移正交，即$\int\phi(t)\phi(t-k)dt=0, k\in Z$。

因此，在$a=2^j$的多分辨率系统中，Haar小波构成一组最简单的正交归一的小波族。

例如，对只有4个像素的一维图像进行Haar小波变换。此图像对应像素值为[11 9 5 7]，用Haar小波变换的过程是计算相邻像素对的平均值，得到一幅分辨率为原图像二分之一的新图像[10 6]。这时，图像信息已经部分丢失，为了能从两个像素组成的图像重构出4个像素的原图像，必须把每个像素对的第一个像素值减这个像素的平均值作为图像的细节系数保存。因此，原图像可以用两个平均值两个细节系数表示：

[10 6 1 −1]。可以把第一步变换得到的图像进一步变换，原图像两级变换过程如下所示。

表7.1 Haar小波变换过程

精　度	平 均 值	细 节 值
4	[11 9 5 7]	
2	[10 6]	[1 −1]
1	[8]	[2]

Haar 小波变换过程实际上是用求平均值和差值的方法对函数或图像进行分解，对于上例，我们可作最多两层的分解。

对于二维图像，同样可以用依次对行列进行小波变换得到二维图像分解方法。这时经过一次小波变换得到的是二维图像的近似值以及水平、垂直和对焦细节分量，显然从二维图像的近似值、水平值、垂直值以及对角细节分量值可以重构出原来的二维图像。

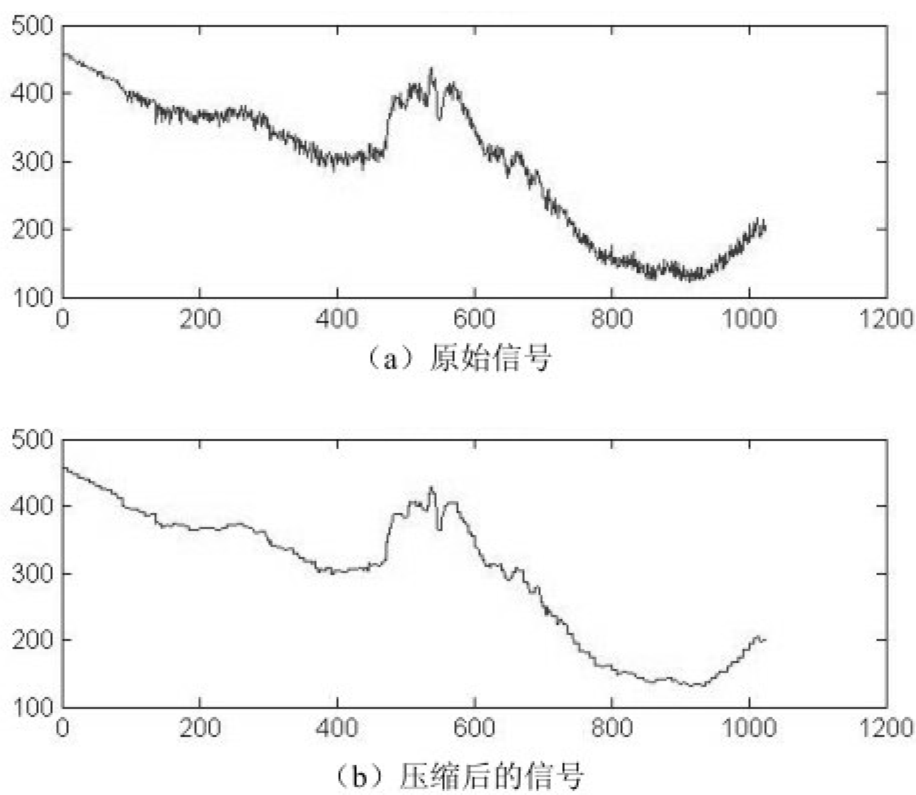
[例7.6] 利用haar小波对信号进行分解并压缩信号

使用函数wdcbm()获取信号压缩阈值，然后采用函数wdencm()实现信号压缩。程序如下。

```
%装载信号
load nelec;
indx=1:1024;
x=nelec(indx);
% 用小波haar对信号进行三层分解
[c,l]=wavedec(x,3,'haar');
alpha=1.5;
% 获取信号压缩的阈值
[thr,nkeep]=wdcbm(c,l,alpha);
% 对信号进行压缩
```

```
[xd,cxd,lxd,perf0,perfl2]=wdencmp('lvd',c,l,'haar',3,thr,'s');
subplot(2,1,1);
plot(indx,x);
title('原始信号');
subplot(2,1,2);
plot(indx,xd);
title('压缩后的信号');
```

程序运行结果如图7.14所示，用Haar小波对信号进行三层分解后，获取信号压缩阈值从而实现压缩的效果。从图7.14中可以看出，压缩后的信号明显比原始信号包含更少的细节信息。



▲图7.14 信号压缩结果

7.3.2 Daubechies小波

Daubechies小波是由世界著名的小波分析学者 Inrid Daubechies构造的小波函数。我们一般写成dbN，N是小波的阶数。小波ϕ(t)和尺度函数φ(t)中的支撑区为2N-1，ϕ(t)的消失矩为N。除N=1外，dbN 不具有对称性（即非线性相位）。dbN没有明确的表达式（除了N=1外），但转换函数h的平方模是很明确的。

令 $P(y)=\sum_{k=0}^{N-1} C_k^{N-1+k} y^k$，其中$C_k^{N-1+k}$为二项式的系数，则有

$$|m_0(w)|^2=(\cos^2\frac{w}{2})^N P(\sin^2\frac{w}{2})$$

式中：$m_0(w)=\frac{1}{\sqrt{2}}\sum_{k=0}^{2N-1} h_k \mathrm{e}^{-\mathrm{j}kw}$。

Daubechies小波具有以下特点。

（1）在时域上是有限支撑的，即ϕ(t)长度有限。而且其高阶原点矩$\int t^p \phi(t)dt=0$，p=0～N；N值越大，ϕ(t)的长度就越长。

（2）在频域上ϕ(w)在w=0处有N阶零点。

（3）ϕ(t)和它的整数位移正交归一，即$\int\phi(t)\phi(t-k)dt=\delta_k$。

（4）小波函数ϕ(t)可以由所谓“尺度函数”φ(t)求出来。尺度函数φ(t)为低通函数，长度有限，支撑域在t=0～2N−1范围内。

[例7.7] 利用db1小波对图像分解和重构

二维离散小波变换只提供了一个函数swt2()，因为它不对分解系数进行下采样，所以单层分解和多层分解的结果是一样的。程序如下。

```
load noiswom
[swa,swh,swv,swd]=swt2(X,3,'db1');
% 使用db1小波对noiswom图像进行三层静态小波分解
whos
```

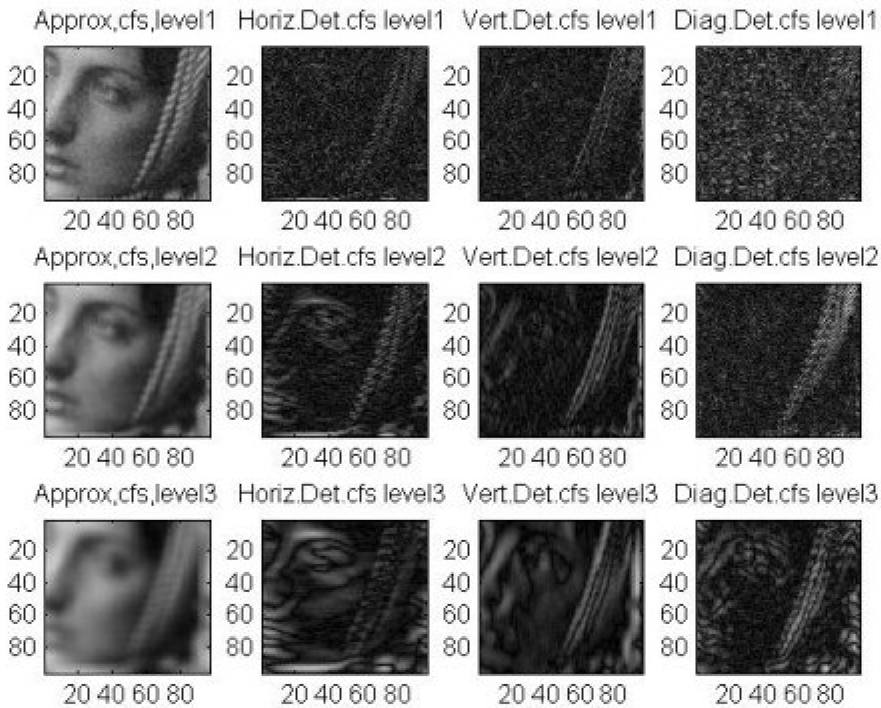
% 可以看出，swt2所小波分解同样不改变信号的长度，原来的96×96的图

% 像做了三层分解以后，分解系数是12个96×96的图像

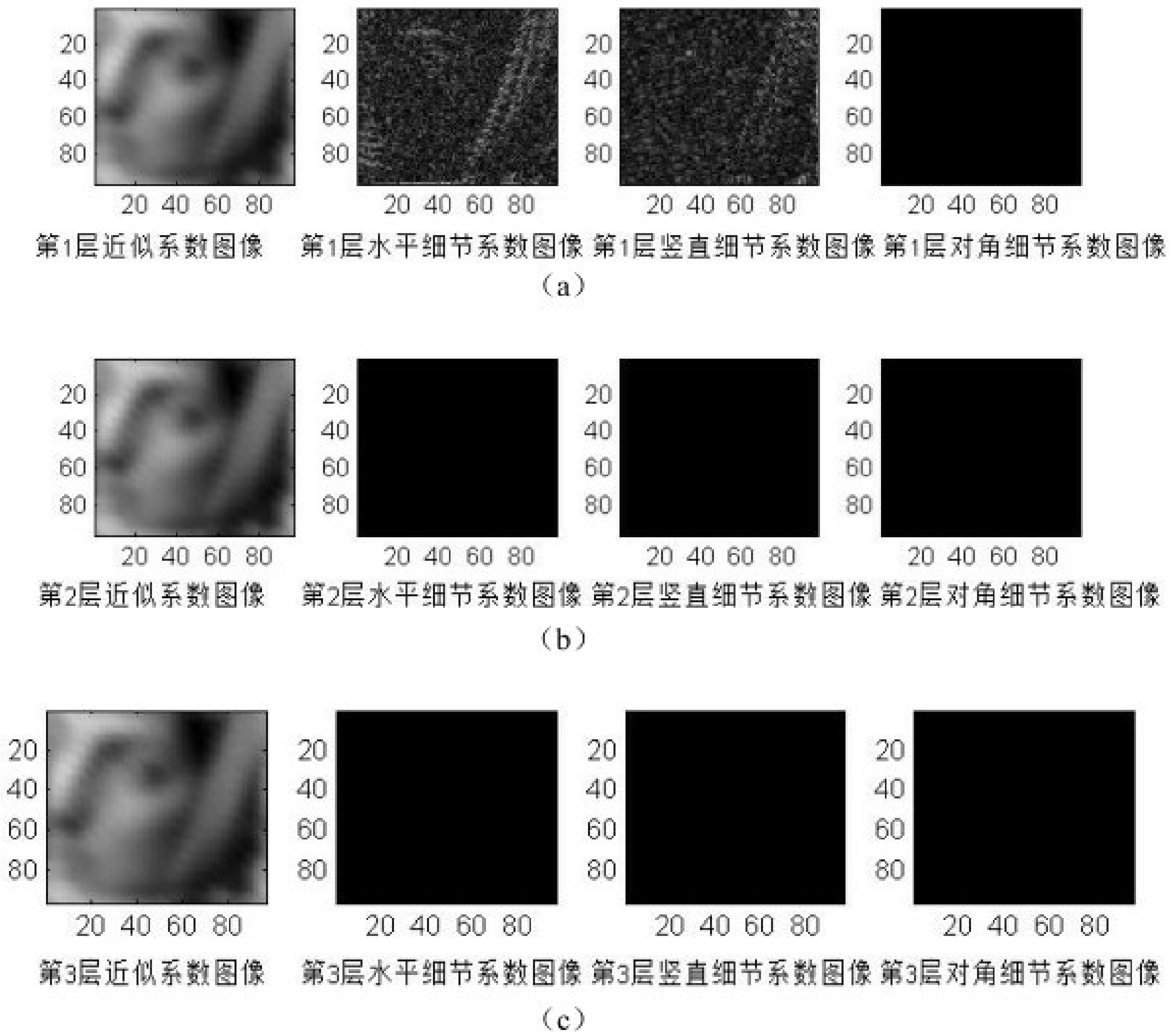
```
colormap(map)
kp=0;
for i=1:3
subplot(3,4,kp+1),image(wcodemat(swa(:,:,i),192));
title(['Approx,cfs,level',num2str(i)])
% 显示第i层近似系数图像，以192字节为单位编码
subplot(3,4,kp+2),image(wcodemat(swh(:,:,i),192));
title(['Horiz.Det.cfs level',num2str(i)])
subplot(3,4,kp+3),image(wcodemat(swv(:,:,i),192));
title(['Vert.Det.cfs level',num2str(i)])
subplot(3,4,kp+4),image(wcodemat(swd(:,:,i),192));
title(['Diag.Det.cfs level',num2str(i)])
kp=kp+4;
end %图像分解结束
%图像重构
load noiswom
[swa,swh,swv,swd]=swt2(X,3,'db1');
% 使用db1小波对noiswom图像进行三层小波分解
mzero=zeros(size(swd));
A=mzero;
A(:,:,3)=iswt2(swa,mzero,mzero,mzero,'db1');
% 使用iswt2的滤波器功能，重建第3层的近似系数，为了避免iswt的合
% 成运算，注意在重建过程中，应保证其他各项系数为零
```

```
H=mzero;V=mzero;D=mzero;
for i=1;3
swcfs=mzero;swcfs(:,:,i)=swh(:,:,i);
H(:,:,i)=iswt2(mzero,swcfs,mzero,mzero,'db1');
swcfs=mzero;swcfs(:,:,i)=swv(:,:,i);
V(:,:,i)=iswt2(mzero,mzero,swcfs,mzero,'db1');
swcfs=mzero;swcfs(:,:,i)=swh(:,:,i);
H(:,:,i)=iswt2(mzero,mzero,mzero,swcfs,'db1');
end
% 分别重建1～3级的各个细节系数，同样在重建某一吸收的时候，要令其他系数为0
A(:,:,2)=A(:,:,3)+H(:,:,3)+V(:,:,3)+D(:,:,3);
A(:,:,1)=A(:,:,2)+H(:,:,2)+V(:,:,2)+D(:,:,2);
% 使用递推的方法建立地1层和第2层近似系数
colormap(map)
kp=0;
for i=1:3
subplot(3,4,kp+1),image(wcodemat(A(:,:,i),192));
title(['第',num2str(i),'层近似系数图像'],'fontsize',6)
subplot(3,4,kp+2),image(wcodemat(H(:,:,i),192));
title(['第',num2str(i),'层水平细节系数图像'],'fontsize',6)
subplot(3,4,kp+3),image(wcodemat(V(:,:,i),192));
title(['第',num2str(i),'层竖直细节系数图像'],'fontsize',6)
subplot(3,4,kp+4),image(wcodemat(D(:,:,i),192));
title(['第',num2str(i),'层对角细节系数图像'],'fontsize',6)
kp=kp+4;
end
```

显示的结果如图 7.15 所示，由于分解过程中没有改变信号的长度，所以在显示近似和细节系数时不需要重建。



▲图7.15 利用db1小波进行图像分解



▲图7.16 利用db1小波对图像进行各级静态小波重建

7.4 高维小波

一维情况下的小波理论具有了丰富的成果，基于小波变换的多分辨率分析，我们知道分解过程实际上需要对滤波后的信号进行抽样，而重建过程需要对子带信号进行插值再滤波。对于一维信号，这类抽样与插值较容易实现，但对于二维和多维情形就复杂得多。由于对多维信号的抽样，实际上是对输入样本的下标进行处理，保留部分样本输出，一般下标处理是用一个抽样矩阵来实现的，即

$$y(K)=x(D\cdot K),K\in Z^2 \tag{7-33}$$

式中：x为输入信号；y为输出信号。抽样矩阵D的行列式的绝对值|det(D)|=N，表示抽样率，即将在 N 个输入样本中抽取一个样本作为输出。在不分离的多分辨率分析中，抽样矩阵为对角阵且各维的特征值均相等，即D=2·I，其中2为维数，I为二维单位矩阵。

多维多分辨率分析中，三维多分辨率分析是最具有实用意义的，这不仅因为三维信息能为我们认识宏观或微观客体提供真实的资料，而且我们每天都面对大量的三维信号加工处理。最典型的三维信号是下面两种情形。

（1）关于时间轴的视频信号或图像序列，每帧图像是空间二维的，而各帧图像之间关系是时序的；对该类三维图像处理的目的，通常是对客体的运动轨迹和运动速度作出估计，或进行序列图像的压缩存储和传输。

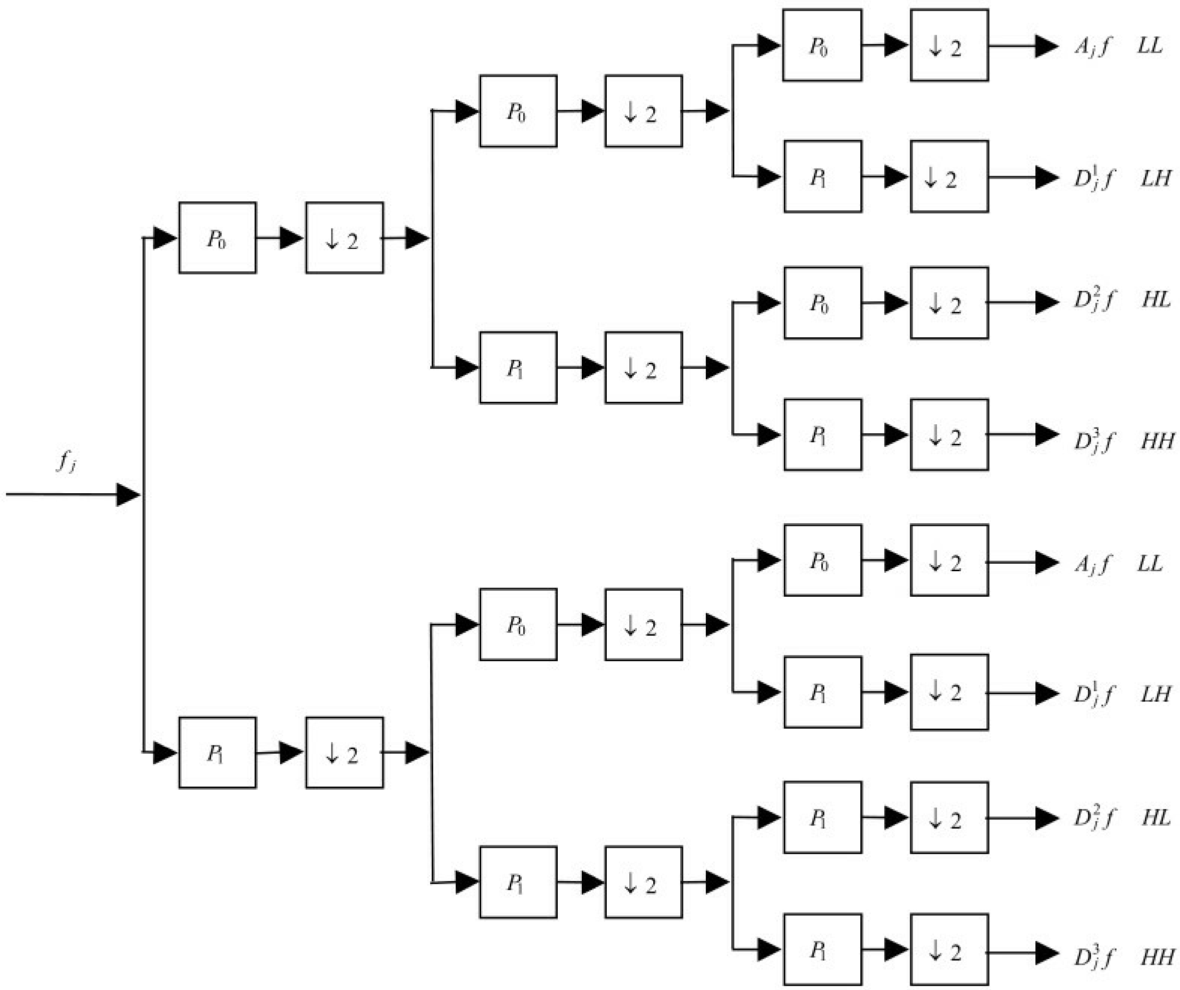
（2）关于z轴计算机断层成像的图像序列，如医用和工业用CT或医用MR等成像设备，它的每帧图像是空间二维的，而各帧图像之间的关系关于z轴是序列的，是真正的三维图像比时间序列的三维图像要复杂得多。对该类三维图像的处理目的通常是要计算客体的三维形状、尺寸和中心位置，或进行序列图像的压缩存储与传输。

一般来说，三维序列图像中，空间维（二维）与时间维（一维）的特性差异性较大，是可以分离来进行多分辨率分析的，而真三维的图像原则上难以分离。幸运的是，我们常面对的是数字序列图像，不管是每帧图像（二维）还是序列图像（三维），都当作可分离情形来处理，实际上均能取得满意的结果，这就为我们的工作带来极大的便利。

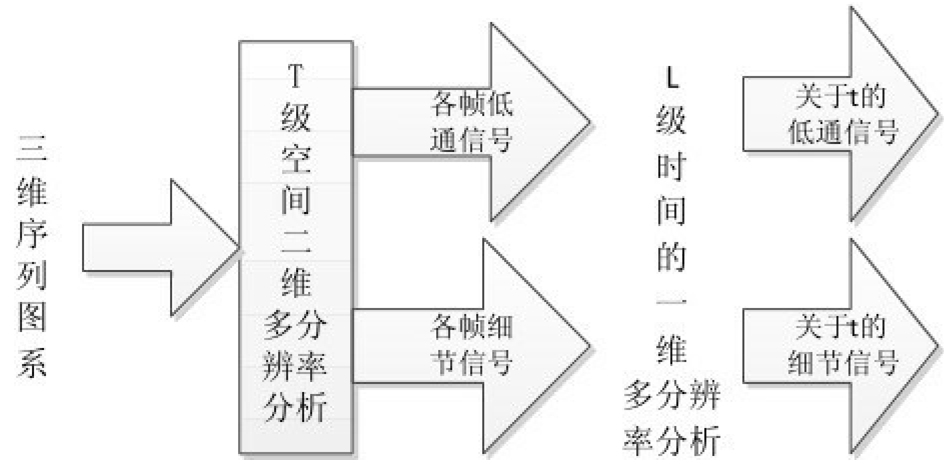
既然三维序列图像，不管哪一种情形，都认为是可分离的，则可以用分解框图给出其中一级的多分辨率分析过程。

（1）关于z轴的三维序列图像的一级多分辨率分析，见图7.17。

（2）关于t轴三维序列图像的一级多分辨率分析，见图7.18。



▲图7.17 关于Z轴可分离三维多分辨率分析（一级）



▲图7.18 关于t轴可分离三维多分辨率分析（一级）

对有限序列信号f(x,y,θ)∈L^2 (r^2 ×θ)，θ=z或t，在闭子空间序列{ V_j^3 }$_{j\in z}$∈L^2 (r^2 ×θ)中满足7.1节中提到的多分辨率分析的定义的五条性质，就说闭子空间序列

$$\{V_j^3\}_{j\in z} \in L^2(R^2\times\theta)$$

是一个三维多分辨率分析。与此相类似，有三维尺度函数

$$\phi(x,y,\theta)=\phi(x)\phi(y)\phi(\theta), \qquad \forall j\in Z$$

其平移系

$$\phi_{j,k_1,k_2,k_3}=\{\phi_{j,k_1}\phi_{j,k_2}\phi_{j,k_3} \mid k_1,k_2,k_3\in Z^3\}$$

构成V_j^3的标准正交基。

它的8个三维小波函数为

$$\varphi^{(1)} = \phi(x)\phi(y)\phi(\theta)$$

$$\varphi^{(2)} = \varphi(x)\phi(y)\phi(\theta)$$

$$\varphi^{(3)} = \phi(x)\varphi(y)\phi(\theta)$$

$$\varphi^{(4)} = \phi(x)\phi(y)\varphi(\theta)$$

$$\varphi^{(5)} = \phi(x)\varphi(y)\varphi(\theta)$$

$$\varphi^{(6)} = \varphi(x)\phi(y)\varphi(\theta)$$

$$\varphi^{(7)} = \varphi(x)\varphi(y)\phi(\theta)$$

$$\varphi^{(8)} = \varphi(x)\phi(y)\varphi(\theta)$$

它们的伸缩平移系

$$\{\varphi_{j,k,l,m}^{(a)} \mid j,k,l,m \in Z^4, a = 1,2,\cdots,8\}$$

构成L^2 ($r^2\times\theta$)的一个标准正交基。其他不再一一表述。

第8章 图像复原

在成像过程中，由于成像系统各种因素的影响，可能使图像质量降低，即“退化”。与图像增强相似，图像复原的目的也是改善图像质量。但是图像复原是试图利用退化过程的先验知识使已被退化的图像恢复本来面目，而图像增强是用某种试探的方式改善图像质量，以适应人眼的视觉与心理。引起图像退化的因素包括由光学系统、运动等造成的图像模糊，以及源自电路和光学因素的噪声等。图像复原是基于图像退化的数学模型，复原的方法也建立在比较严格的数学推导上。因此，有较多而复杂的数学运算是本章的特点。

本章的知识和技术热点

- 图像复原的基本概念
- 图像复原的一般模型
- 实用图像复原技术

本章的典型案例分析

- 利用图像复原方法复原模糊噪声图像

8.1 图像复原的一般理论

图像复原技术的目的是使退化了的图像尽可能恢复到原来的真实面貌，所以对其理论的了解显得尤为重要。本节主要讲述图像复原的基本概念、基本模型，以及常用的几种噪声模型，在介绍完这些的基础上，继续了解一下常用的比较简单的线性一维不变模型。

8.1.1 图像复原的基本概念

图像复原与第5章介绍的图像增强相似，两者都是要得到在某种意

义上改进的图像，或者说，希望改进输入图像的质量。两者不同之处是图像增强技术一般要借助人的视觉系统的特性，以取得看起来较好的视觉效果。而图像复原则认为图像在某种情况下退化或恶化了（图像品质下降了），现在需要根据相应的退化模型和知识重建或恢复原始的图像。尽管两者有相交叉的领域，但图像增强主要是一个主观的过程，而图像复原的大部分过程是一个客观的过程。也就是说，图像复原技术是将图像退化的过程模型化，并据此采取相反的过程以得到原始的图像。

图像复原的前提是图像退化，图像退化是指图像在形成、记录、处理、传输过程中由于成像系统、记录设备、处理方法和传输介质的不完善，导致的图像质量下降。具体来说，常见的退化原因大致有成像系统的像差或有限孔径或存在衍射、成像系统的离焦、成像系统与景物的相对运动、底片感光特性曲线的非线性、显示器显示时的失真、遥感成像中的大气散射和大气扰动、遥感摄像机的运动和扫描速度不稳定、系统各个环节的噪声干扰、模拟图像数字化引入的误差等。

将损失掉的图像质量部分复原过来，可以起到不同的作用：可能像化妆品一样只起修饰作用，也可能起到成败攸关的作用，例如太空研究中月球和行星图像的获取。图像复原可以抽象的表述为退化过程可以看作是一个变换T，若原始图像为f(x,y)，退化图像为g(x,y)，则有$T[f(x,y)]\rightarrow g(x,y)$。复原就是由g(x,y)求f(x,y)，即进行逆变换t^{-1}，使得$t^{-1}[g(x,y)]\rightarrow f(x,y)$。

8.1.2 图像复原的一般模型

对于退化的复原，一般可采用两种方法。一种方法适用于对图像缺乏已知信息的情况，此时可对退化过程（模糊和噪声）建立模型，进行描述，并进而寻找一种去除或削弱其影响的过程。由于这种方法试图估计图像被一些相对良性的退化过程影响以前的情况，故是一种估计方

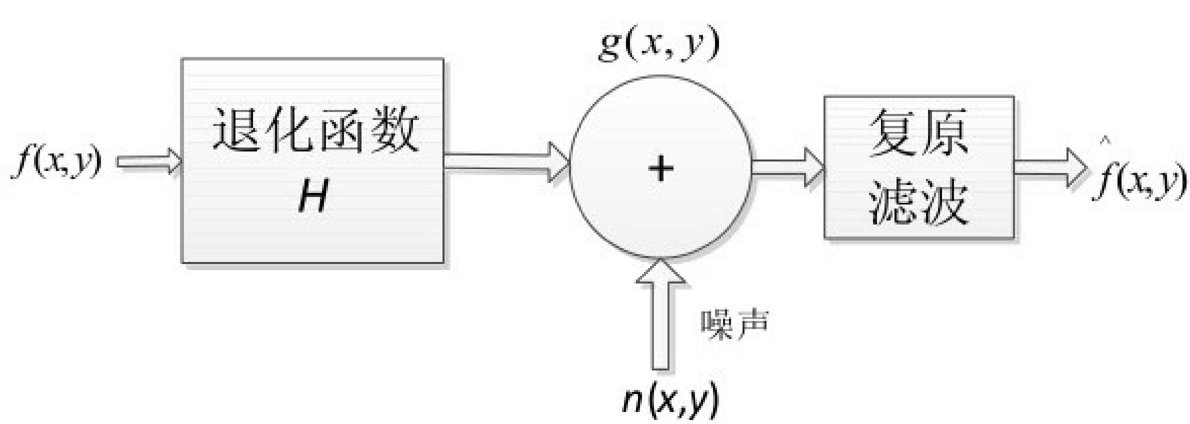
法。

另一方面，若对于原始图像有足够的已知信息，则对原始图像建立一个数学模型并根据它对退化图像进行拟合会更有效。例如，假设已知图像中仅含有确定大小的圆形物体（如星辰、颗粒、细胞等），这样，由于仅是原始图像很少的几个参数（数目、位置、幅度等）未知，因此这是一个检测问题。

在进行图像复原时，还有许多其他选择。首先，问题既可以用连续数学，也可以用离散数学进行处理。其次，处理既可以在空间域，也可以在频域进行。此外，当复原必须用数字方法进行时，处理既可以通过空间域卷积，也可以通过频域的相乘来实现。

1．图像退化复原模型

本节将用图8.1所示的系统作为图像的退化和复原的模型。退化过程可以被模型化为一个退化函数和一个加性噪声项，对一幅输入图像f(x,y)进行处理，产生一幅退化图像g(x,y)。给定g(x,y)和关于退化函数H的一些信息以及外加噪声项n(x,y)，图像复原的目的是获得关于原始图像的近似估计$\hat{f}$(x,y)。通常我们希望这一估计尽可能接近原始输入图像，并且H和n的信息知道的越多，所得到的$\hat{f}$(x,y)就会越接近f(x,y)。



▲图8.1 图像退化和复原模型图

如果系统H是一个线性、位置不变性的过程，那么在空间域中给出的退化图像可由下式给出。

$$g(x,y)=h(x,y)*f(x,y)+n(x,y) \tag{8-1}$$

式中：h(x, y)是退化函数的空间描述；“*”表示空间卷积。由于空间域上得卷积等同于频域上得乘积，因此式（8-1）的模型的等价频域描述如下。

$$G(u,v)=H(u,v)F(u,v)+N(u,v) \tag{8-2}$$

公式中的大写字母项是式（8-1）中相应项的傅立叶变换。

2．噪声模型

数字图像的噪声主要来源于图像的获取和传输过程。图像传感器的工作情况受各种因素的影响，如图像获取中的环境条件和传感元器件自身的质量。例如，当使用CCD摄像机获取图像时，光照强度和传感器的温度是生成图像中产生大量噪声的主要因素。图像在传输过程中主要由于所用传输信道被干扰而受到噪声污染。例如，通过无线网络传输的图像可能会因为光或其他大气因素的干扰被污染。下面介绍一些重要噪声的概率密度函数。

（1）高斯噪声

在空间域和频域中，由于高斯噪声在数学上的易处理性，这种噪声模型经常被用于实践中。事实上，这种非常方便的易处理性，使得高斯模型经常用于临界情况下。

高斯随机变量z的概率密度函数由下式给出。

$$p(z)=\frac{1}{\sqrt{2\pi}\sigma}\mathrm{e}^{-(z-\mu)^2/2\sigma^2} \tag{8-3}$$

式中：z表示灰度值；μ表示z的平均值或期望值；σ表示z的标准差。标准差的平方σ^2称为z的方差。

（2）瑞利噪声

瑞利噪声的概率密度函数由下式给出

$$p(z)=\begin{cases}\dfrac{2}{b}(z-a)\mathrm{e}^{-(z-a)^{2/b}} & z\geqslant a\\ 0 & z<a\end{cases} \qquad (8\text{-}4)$$

概率密度的均值和方差由

$$\mu=a+\sqrt{\pi b/4} \qquad (8\text{-}5)$$

和

$$\sigma^2=\frac{b(4-\pi)}{4} \qquad (8\text{-}6)$$

给出。

（3）伽马噪声

伽马噪声的概率密度函数由下式给出。

$$p(z)=\begin{cases}\dfrac{a^b\ z^{b-1}}{(b-1)!}\mathrm{e}^{-az} & z\geqslant 0\\ 0 & z<0\end{cases} \qquad (8\text{-}7)$$

其中，a>0,b为正整数且"！"表示阶乘。其密度的均值和方差由

$$\mu=\frac{b}{a} \qquad (8\text{-}8)$$

和

$$\sigma^2 = \frac{b}{a^2} \qquad (8\text{-}9)$$

给出。

（4）指数分布噪声

指数噪声的概率密度函数可由下式给出。

$$p(z) = \begin{cases} a\mathrm{e}^{-az} & z \geqslant 0 \\ 0 & z < 0 \end{cases} \qquad (8\text{-}10)$$

其中，a>0。概率密度函数的期望值和方差是

$$\mu = \frac{1}{a} \qquad (8\text{-}11)$$

和

$$\sigma^2 = \frac{1}{a^2} \qquad (8\text{-}12)$$

注意，指数分布的概率密度函数是当b=1时伽马概率密度分布的特殊情况。

（5）均匀分布噪声

均匀分布噪声的概率密度可由下式给出。

$$p(z) = \begin{cases} \dfrac{1}{b-a} & b \geqslant z \geqslant a \\ 0 & z < 0 \end{cases} \qquad (8\text{-}13)$$

概率密度函数的期望值和方差可由下式给出。

$$\mu = \frac{a+b}{2} \tag{8-14}$$

$$\sigma^2 = \frac{(b-a)^2}{12} \tag{8-15}$$

（6）脉冲噪声（椒盐噪声）

（双极）脉冲噪声的概率密度函数可由下式给出。

$$p(z) = \begin{cases} P_a & z = a \\ P_b & z = b \\ 0 & 其他 \end{cases} \tag{8-16}$$

如果b>a，则灰度值b在图像中将显示为一个亮点；反之则a的值将显示为一个暗点。若p_a或p_b为零，则脉冲噪声称为单极脉冲。如果p_a和p_b均不可能为零，尤其是它们近似相等时，则脉冲噪声值将类似于随机分布在图像上的胡椒和盐粉微粒。由于这个原因，双极脉冲噪声也称为椒盐噪声。同时，它们有时也成为散粒和尖峰噪声。

噪声脉冲可以是正的，也可以是负的。标定通常是图像数字化过程的一部分。与图像信号的强度相比，脉冲干扰通常较大，因此在一幅图像中，脉冲噪声总是数字化为最大值（纯黑或纯白）。这样，通常假设a和b是饱和值，从某种意义上看，在数字化图像中，它们等于所允许的最大值和最小值。由于这一结果，负脉冲以一个黑点出现在图像中。由于相同的原因，正脉冲以白点出现在图像中。

上述的一组概率密度函数为在实践中模型化宽带噪声干扰状态提供了有用的工具。例如，在一幅图像中，高斯噪声的产生源于电子电路噪声和由低照明或高温带来的传感器噪声。瑞利密度分布在图像范围内特征化噪声现象时非常有用。指数密度分布和伽马密度分布在激光成像中

有一些应用。脉冲噪声主要表现在成像中的短暂停留，例如错误的开关操作。均匀密度分布可能在实践中描述的最少，然而均匀密度作为模拟随机数产生器的基础是非常有用的。

[例8.1]几种噪声的随机数据直方图

为了能详细理解以上几种噪声，并了解其使用，这里利用函数imnoise2()产生上述几种噪声模式本身。我们可以直接指定需要的噪声参数。该函数产生一个大小为L×H的噪声数组M，为了使这个数组有用，需要对它进行进一步的处理。例如，若要用这个数组来污染一幅图像，我们可以使用函数find寻找M中所有值为0的坐标，并把图像中相应的坐标置为可能的最小灰度值。下面的例子是使用该函数产生几种噪声的类型的随机数据直方图。

function M = imnoise2(type, L, H, x, y)

% IMNOISE2函数用以产生前面所述几种噪声的随机序列.

% M = IMNOISE2(TYPE, L, H, x, y) 这里用M来代替该随机序列数组，

% 该数组是L×H维, 里面的元素都是根据上面的噪声类型相应产生的

% x和y是它们的相关参数。

% 只有当需要的噪声类型包含在此函数中时，才能产生相应的随机数组

% 而且需要提供L与H的值，才能确定该数组的大小

% 举例说明如下：如果L=H=1，则该函数产生特定类型的具有参数x，y的单个序列数

% 我们规定的噪声类型和参数x、y参数如下： %

% 均匀噪声　　均匀噪声序列的相关参数是（x，y）

%　默认值是　(0, 1).

% 高斯噪声　　高斯序列噪声的参数是平均值x和均方差y

```
%    默认值是    x = 0, y = 1.
% 椒盐噪声       当幅值是0时，Pa = x 幅值是1时，Pb=y
%                默认的值是Pa = Pb = x = y = 0.05
%                当幅值不是0或1时，矩阵M就和三个参数有关。
%                如果M(x, y) =0，噪声在（x,y）处就是胡椒噪声，如果
%                M(x, y) = 1，噪声就是盐粒噪声
%                如果M(x, y) = 0.5, 就没有噪声
% 对数噪声       对数噪声的参数是偏置x和形状参数y
%                默认值是x = 1 and y =0.25.
% 瑞利噪声       默认参数是 x = 0 和 y = 1.
% 指数噪声       指数噪声只有一个参数，默认值是A = 1.
% 伽马噪声       该噪声有两个参数，其默认值是A = 2 and B = 5.
% 设置默认值
if nargin == 1
  x = 0; y = 1;
  L = 1; H = 1;
elseif nargin == 3
  x = 0; y = 1;
end
% 开始处理
switch lower(type)
%均匀噪声的情况
case 'uniform'
  M = x + (y – x)*rand(L, H);
%高斯噪声的情况
case 'gaussian'
  R = a + b*randn(M, N);
```

```
%椒盐噪声的情况
case 'salt & pepper'
   if nargin <= 3
      x= 0.05; y = 0.05;
   end
   % 确认Pa + Pb 是否大于1.
   if (x+ y) > 1
      error('The sum Pa + Pb must not exceed 1.')
   end
R(1:L, 1:H) = 0.5;
% 不是0或1的情况
N = rand(L, H);
c = find(N <= a);
R(c) = 0;
u = a + b;
c = find(N > a & N <= u);
M(c) = 1;
%对数噪声的情况
case 'lognormal'
   if nargin <= 3
      a = 1; b = 0.25;
   end
M = a*exp(b*randn(L, H));
%瑞利噪声的情况
case 'rayleigh'
   M = a + (-b*log(1 - rand(L, H))).^0.5;
%指数噪声的情况
```

```
case 'exponential'
  if nargin <= 3
    a = 1;
  end
  if a <= 0
    error('Parameter a must be positive for exponential type.')
  end
  k = -1/a;
  M = k*log(1 - rand(L, H));
%伽马噪声的情况
case 'erlang'
  if nargin <= 3
    a = 2; b = 5;
  end
  if (b ～= round(b) | b <= 0)
    error('Param b must be a positive integer for Erlang.')
  end
  k = -1/a;
  R = zeros(M, N);
  for j = 1:b
    R = R + k*log(1 - rand(M, N));
  end
otherwise
  error('Unknown distribution type.')
end
%下面的程序利用函数imnoise2()来显示上面几种噪声的随机直方图
r=imnoise2('gaussian',100000,1,0,1); %产生参数是x=0，y=1的高斯噪
```

声序列

subplot(3,2,1);

hist(r,50); %显示直方图，即（8.2）的（a）图

r=imnoise2('rayleigh',100000,1,0,1); %产生参数是x=0，y=1的瑞利噪声序列

subplot(3,2,2);

hist(r,50); %显示直方图，即（8.2）的（b）图

r=imnoise2('lognormal',100000,1,1,0.25); %产生参数是x=1，y=0.25的对数噪声序列

subplot(3,2,3);

hist(r,50); %显示直方图，即（8.2）的（c）图

r=imnoise2('exponential',100000,1,1); %产生参数是x=1，y=1的指数噪声序列

subplot(3,2,4);

hist(r,50); %显示直方图，即（8.2）的（d）图

r=imnoise2('uniform',100000,1,0,1); %产生参数是x=0，y=1的均匀噪声序列

subplot(3,2,5);

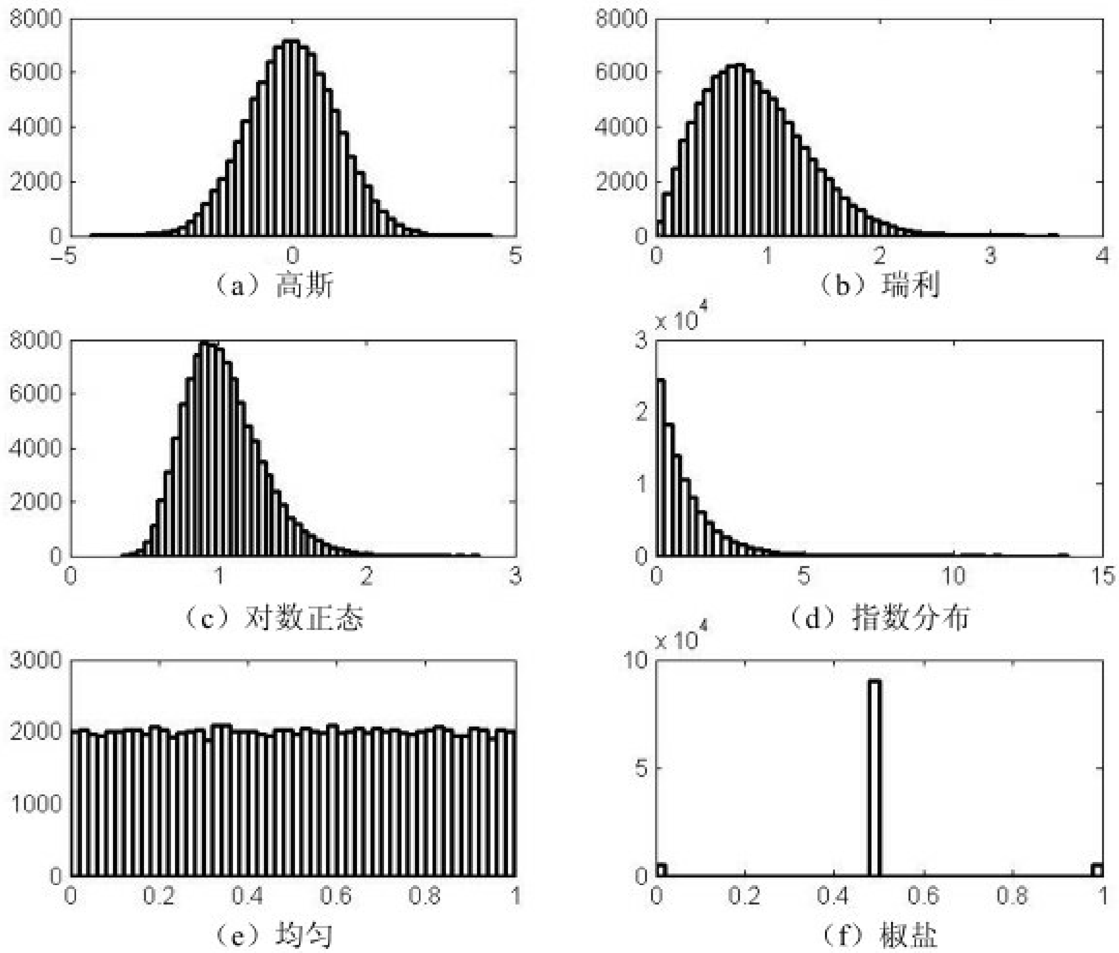
hist(r,50); %显示直方图，即（8.2）的（e）图

r=imnoise2('salt & pepper',100000,1,0.05,0.05); %产生参数是x=0.05，y=0.05的椒盐噪声序列

subplot(3,2,6);

hist(r,50); %显示直方图，即（8.2）的（f）图

每幅图都产生一个有着 100 000个元素的列向量，产生的直方图如图 8.2所示，图 8.2（a）中每个元素都来自均值为0、标准偏差为1的高斯分布的随机数。同理，其他图是相应的噪声模式的直方图。



▲图8.2 随机数的直方图

3．空间滤波复原

当在一幅图像中唯一存在的退化是噪声时，式（8-1）和式（8-2）变成

$$g(x,y)=f(x,y)+n(x,y) \tag{8-17}$$

和

$$G(u,v)=F(u,v)+N(u,v) \tag{8-18}$$

由于噪声项是未知的，我们不能从g(x,y)或G(u,v)中减去它们。而如果在周期噪声的情况下，通常从G(u,v)的谱来估计N(u,v)是可能的。

在只有加性噪声存在的前提下，我们选择空间滤波方法。下面介绍几种常见的用于减少噪声的空间滤波器。

（1）均值滤波器

在图像上，对待处理的像素给定一个模板，该模板包括了其周围的邻近像素。用模板中的全体像素的均值来替代原来的像素值的方法就是均值滤波。它主要包括算术均值滤波、几何均值滤波、谐波均值滤波以及逆谐波均值滤波。

由于均值滤波器对所有的点都是同等对待，所以在将噪声点分摊的同时，将景物的边界点也分摊了，会使图像变得模糊。为改善其效果，可以采用加权平均的方式来构造滤波器。但是，它所滤波的效果也是有限的。为有效改善这种状况，须改换滤波器的设计思路，统计排序滤波就是一种比较有效的方法。

（2）统计排序滤波器

统计排序滤波器是一种非线性的空间滤波器，它的响应基于图像滤波器包围的图像区域中像素的排序，然后用统计排序结果决定的值代替中心像素的值。统计滤波器最常见的例子是中值滤波器，此滤波器我们在前述5.4节中已经详细介绍过，这里就不再介绍了。

[例8.2]图像空域滤波

利用编写的函数split()对图像进行空域滤波，此函数既可以进行线性滤波，也可以进行非线性空域滤波，并且概括了多种滤波方式，如中值滤波、最值滤波、均值滤波以及调和滤波等。由于图像增强部分已经介绍过相应的滤波函数，并利用这几种滤波器进行了滤波。因此，此处我们在将一幅图像分别利用胡椒噪声和盐粒噪声污染后，利用不同Q值的反调和滤波器来滤波并观察滤波效果。

%用f来描述函数

```
function f = spfilt(g, type, m, n, parameter)
%我们所设计的滤波器主要包括以下几种：
%   F = SPFILT(G, 'amean', M, N)       算术平均数滤波
%   F = SPFILT(G, 'gmean', M, N)       几何均值滤波
%   F = SPFILT(G, 'hmean', M, N)       谐波均值滤波
%   F = SPFILT(G, 'chmean', M, N, Q)   对比谐波均值滤波
%                                默认的滤波阶数 Q = 1.5.
%   F = SPFILT(G, 'median', M, N)      中值滤波
%   F = SPFILT(G, 'max', M, N)         最大值滤波
%   F = SPFILT(G, 'min', M, N)         最小值滤波
%   F = SPFILT(G, 'midpoint', M, N)    中点滤波
%   F = SPFILT(G, 'atrimmed', M, N, D) 修剪均值滤波.
%                                参数D须是一个非负数
%                                默认值是 D = 2.
%
% 默认值 M = N = 3,
% Q = 1.5, and D = 2.
% 处理输入
if nargin == 2
  m = 3; n = 3; Q = 1.5; d = 2;
elseif nargin == 5
  Q = parameter; d = parameter;
elseif nargin == 4
  Q = 1.5; d = 2;
else
  error('Wrong number of inputs.');
end
```

```
% 开始滤波
switch type
%算术平均滤波的情况
case 'amean'
  w = fspecial('average', [m n]);
  f = imfilter(g, w, 'replicate');
%几何均值滤波的情况
case 'gmean'
f = gmean(g, m, n);
%调和滤波的情况
case 'hmean'
  f = harmean(g, m, n);
%反调和滤波的情况
case 'chmean'
  f = charmean(g, m, n, Q); %利用相应的函数
%中值滤波的情况
case 'median'
  f = medfilt2(g, [m n], 'symmetric');
最大值滤波的情况
case 'max'
  f = ordfilt2(g, m*n, ones(m, n), 'symmetric');
最小值滤波的情况
case 'min'
  f = ordfilt2(g, 1, ones(m, n), 'symmetric');
%中点滤波的情况
case 'midpoint'
  f1 = ordfilt2(g, 1, ones(m, n), 'symmetric');
```

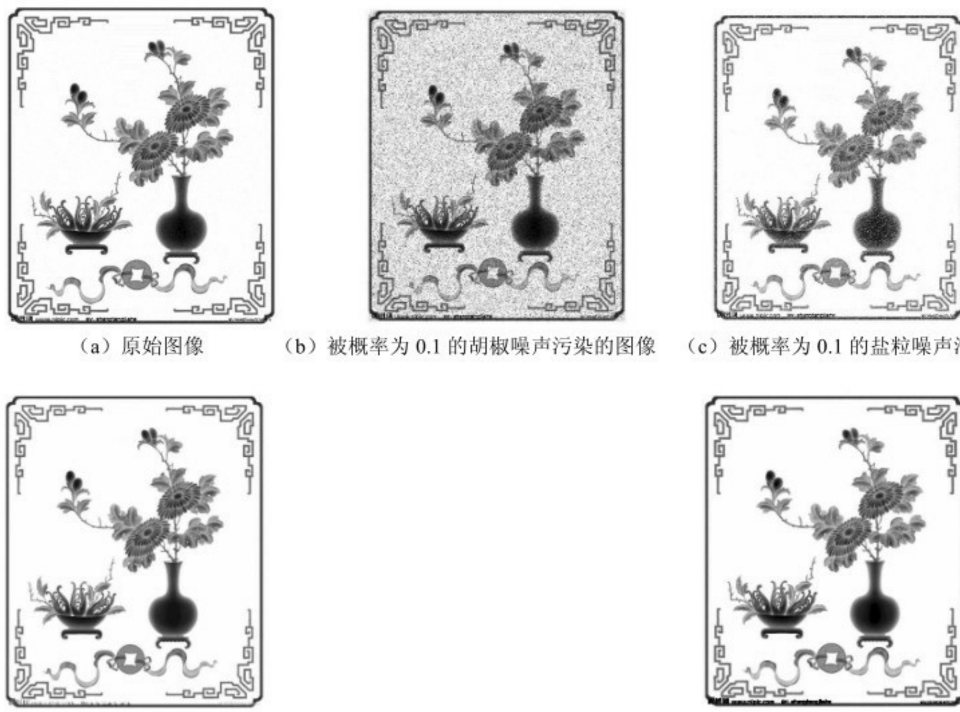
```
  f2 = ordfilt2(g, m*n, ones(m, n), 'symmetric');
  f = imlincomb(0.5, f1, 0.5, f2);
%修剪滤波的情况
case 'atrimmed'
  if (d <= 0) | (d/2 ～= round(d/2))
     error('d must be a positive, even integer.')
  end
  f = alphatrim(g, m, n, d);
%其他
otherwise
  error('Unknown filter type.')
end
%下面是一些具体的函数，实现相应的滤波
function f = gmean(g, m, n)
% 设置一个几何均值滤波器
inclass = class(g);
g = im2double(g);
warning off;
f = exp(imfilter(log(g), ones(m, n), 'replicate')).^(1 / m / n);
warning on;
f = changeclass(inclass, f);
%------------------------------------------------------------------%
function f = harmean(g, m, n)
% 设置调和滤波器
inclass = class(g);
g = im2double(g);
f = m * n ./ imfilter(1./(g + eps),ones(m, n), 'replicate');
```

```
f = changeclass(inclass, f);
%-----------------------------------------------------------------%
function f = charmean(g, m, n, q)
% 设置反调和滤波器
inclass = class(g);
g = im2double(g);
f = imfilter(g.^(q+1), ones(m, n), 'replicate');
f = f ./ (imfilter(g.^q, ones(m, n), 'replicate') + eps);
f = changeclass(inclass, f);
%-----------------------------------------------------------------%
function f = alphatrim(g, m, n, d)
% 设置修剪滤波器
inclass = class(g);
g = im2double(g);
f = imfilter(g, ones(m, n), 'symmetric');
for k = 1:d/2
  f = imsubtract(f, ordfilt2(g, k, ones(m, n), 'symmetric'));
end
for k = (m*n - (d/2) + 1):m*n
  f = imsubtract(f, ordfilt2(g, k, ones(m, n), 'symmetric'));
end
f = f / (m*n - d);
f = changeclass(inclass, f);
%下面的程序通过读入一幅图像并将其污染，来利用前面的调和滤波器滤波
%从内存中读入一幅图像
f=imread('D:\3.jpg');
```

```
subplot(3,2,1);
imshow(f);%显示该图像
[M,N]=size(f);
R=imnoise2('salt & pepper',M,N,0.1,0);%将该图像被概率为0.1的胡椒噪声污染
c=find(R==0);
gp=f;
gp(c)=0;
subplot(3,2,2);
imshow(gp);%显示被污染的图像
R=imnoise2('salt & pepper',M,N,0,0.1);%将该图像被盐粒污染
c=find(R==1);
gs=f;
gs(c)=255;
subplot(3,2,3);
imshow(gs)%显示被盐粒污染的图像
fp=spfilt(gp,'chmean',3,3,1.5);%使用Q为正值的反调和滤波器过滤胡椒噪声
subplot(3,2,4);
imshow(fp);
fs=spfilt(gs,'chmean',3,3,-1.5);%使用Q为负值的反调和滤波器滤除盐粒噪声
subplot(3,2,5);
imshow(fs);
```

运行结果如图8.3所示，图8.3（a）是原始图像，图8.3（b）是该幅图像被概率只有0.1的胡椒噪声污染后的图像，图8.3（c）是仅被盐粒噪声污染的图像，过滤胡椒噪声的较好办法是使用Q值为正的反调和滤波

器，图 8.3（d）即是滤波后的图像；而过滤盐粒噪声可以使用 Q 值为负的滤波器，如图8.3（c）所示。



（a）原始图像　（b）被概率为 0.1 的胡椒噪声污染的图像　（c）被概率为 0.1 的盐粒噪声污染的图像

（d）用阶为 Q=1.5 的 3×3 反调和滤波器对（b）滤波的结果　（e）用阶为 Q=-1.5 的 3×3 反调和滤波器对（c）滤波的结果

▲图8.3

4．线性空间不变退化模型

由前述可知，复原前，图8.1中的输入输出关系可以表示为式（8-17）的形式。现在，假设n(x,y)=0，则g(x,y)=H[f(x,y)]。若

$$H[af_1(x,y)+bf_2(x,y)]=aH[f_1(x,y)]+bH[f_2(x,y)] \tag{8-19}$$

则系统H是一个线性系统。这里，a和b是比例常数，$f_1(x,y)$和$f_2(x,y)$是任意两幅输入的图像。

若a=b=1，则式（8-19）变为

$$H[f_1(x,y)+f_2(x,y)]=H[f_1(x,y)]+H[f_2(x,y)] \tag{8-20}$$

这就是所谓的加性。这一特性简单地表明，如果H为线性算子，那么两个输入之和的响应等于两个响应之和。

如果 $f_2(x, y)=0$ ，则式（8-19）变为

$$H[af_1(x,y)] = aH[f_1(x,y)] \tag{8-21}$$

这就是均匀性。它表明任何与乘数相乘的输入的响应等于该输入响应乘以相同的常数，即一个线性算子具有加性和均匀性。

对于任意 f (x, y),α和β，如果

$$H[f(x-\alpha, y-\beta)] = g(x-\alpha, y-\beta) \tag{8-22}$$

则一个具有输入输出关系g(x, y)=H[f (x, y)]的系统称为空间不变系统。这个定义说明图像中任一点的响应只取决于在该点的输入值，而与该点的位置无关。

图像复原中，许多退化类型可近似表示为线性空间不变过程。这一方法的优点在于广泛的线性系统理论工具对于解决图像复原问题很实用。与位置有关的非线性技术虽然更普遍，但是它们会带来很多困难，常常没有解，或者解决计算问题时非常困难。由于退化模型为卷积的结果，并且图像复原需要滤波器，所以图像复原又称为图像去卷积。

20世纪60年代中期，去卷积（逆滤波）开始被广泛应用于数字图像复原。Nathan用二维去卷积方法来处理由漫游者、探索者等外星探索发射得到的图像。由于与噪声相比，信号的频谱随着频率升高下降较快，因此高频部分主要是噪声。Nathan采用的是限定去卷积传递函数最大值的方法。

在同一时期，Harris则采用点扩散函数的解析模型对望远镜图像中由于大气扰动所造成的模糊进行了去卷积处理，Mcglamery采用由实验确定的点扩散函数来对大气扰动去卷积。从那以后，去卷积就成了图像

复原的一种标准技术。

（1）维纳去卷积

在大部分图像中，邻近的像素是高度相关的，而距离较远的像素相关性较弱。由此，可以认为典型图像的自相关函数通常随着与原点的距离增加而下降。由于图像的功率谱是其自相关函数的傅立叶变换，从而可以认为图像的功率谱随着频率的升高而下降。

一般地，噪声源往往具有平坦的功率谱，即使不是如此，其随频率升高而下降的趋势也要比典型的图像功率谱慢得多。因此，可以料想功率谱的低频部分以信号为主，而高频部分则主要被噪声所占据。由于去卷积滤波器的幅值通常随着频率的升高而升高，因此会增强高频处的噪声。早期的去卷积多采用就事论事结合直接判断的方法来处理噪声问题。

Helstrom采用最小均方差估计法，提出了具有如下二维传递函数的维纳去卷积滤波器。

$$G(u,v)=\frac{H^*(u,v)P_f(u,v)}{|H(u,v)|^2\ P_f(u,v)+P_n(u,v)} \tag{8-23}$$

它也可写做

$$G(u,v)=\frac{H^*(u,v)}{|H(u,v)|^2+P_n(u,v)/P_f(u,v)} \tag{8-24}$$

式中：p_f和p_a分别为信号和噪声的功率谱。

Slepian将维纳去卷积推广用于处理随机点扩散函数的情况（如由于大气扰动引起的）。其后，Pratt和Habib提出了提高维纳去卷积计算效率的方法。

维纳去卷积提供了一种在有噪声的情况下导出去卷积传递函数的最

优方法，但有3个问题限制了它的有效性。

首先，当图像复原的目的是供人观察时，均方误差准则并不是一个特别好的优化准则。这是因为此准则对所有误差（不管其在图像中的位置）都赋予同样地权，而人眼则对暗处和高梯区域的误差比其他区域的误差具有较大的容忍性。由于使均方误差最小化，维纳滤波器以一种并非适合人眼的方式对图像进行了平滑。

其二，经典的维纳去卷积不能处理具有空间可变点扩散函数的情形（例如存在彗差、散差、表面像场弯曲以及含有旋转的运动模糊等情况下）。

最后，这种技术不能处理有着非平稳信号和噪声的一般情形。大多数图像都是高度非平稳的，有着被陡峭边缘分开的大块平坦区域。此外，许多重要的噪声源是与局部灰度有伏案的（信号有关噪声）。

（2）功率谱均衡

Cannon证明，如下形式的滤波器可将退化图像的功率谱复原至其原先的幅度。

$$G(u,v)=\left[\frac{P_f(u,v)}{|H(u,v)|^2+P_f(u,v)+P_n(u,v)}\right]^{1/2} \qquad (8\text{-}25)$$

和维纳滤波器类似，这种功率谱均衡（点扩散函数）滤波器也是无相移的（实偶函数），它可用于无相移的或相移可用其他方法确定的模糊传函数。

点扩散函数滤波器式（8-25）与维纳滤波器间的相似是十分明显的。当无噪声时，这两处滤波器都简化为直接的去卷积；当无信号时，这两种滤波器都完全截止。然而，不同的是点扩散函数滤波器在模糊传递函数F(u,v)为零处并不截止到零。

点扩散函数滤波器具有相当强的图像复原能力，在某些情况下其性

能优于维纳滤波器。点扩散函数滤波器有时也被叫做同态滤波器。

（3）几何均值滤波器

考察如下形式的复原滤波器传递函数。

$$G(u,v)=[\frac{H^*(u,v)}{|H(u,v)|^2}]^a[\frac{H^*(u,v)}{|H(u,v)|^2+\gamma P_n(u,v)/P_f(u,v)}]^{1-a} \qquad (8\text{-}26)$$

式中：a和γ为正的实常数。这种滤波器是前面讨论过的几种滤波器的一般形式，其传递函数具有参数a和γ。注意，当a=1时，式（8-26）就成为去卷积滤波器，而若令a=1/2,γ=1，则它就变为式（8-15）形式的功率均衡滤波器。

进一步观察，还可注意到a=1/2时，式（8-26）定义的是普通去卷积和维纳去卷积的几何平均。因此式（8-25）定义的滤波器的第三个名字叫做几何均值滤波器。然而，通常将式（8-26）所定义的更一般的滤波器称为几何均值滤波器。

若式（8-26）中的a=0，就得到了参数化维纳滤波器。

$$G(u,v)=[\frac{H^*(u,v)}{|H(u,v)|^2+\gamma P_n(u,v)/P_f(u,v)}] \qquad (8\text{-}27)$$

当γ=1时，它就变成了式（8-25）形式的维纳去卷积滤波器；而当γ=0时，它就变成了单纯的去卷积。一般来说，可通过选择γ的数值来获得所希望的维纳平滑效果。

式（8-26）代表一个很一般的复原滤波器，可用于具有线性、空间不变的模糊函数和加性、不相关噪声的情形。Andrews 和 Hunt 对在轻微模糊的适度噪声条件下式（8-26）所定义的滤波器的复原能力进行了研究。其结果表明，在上述条件下，单纯去卷积的效果最差；维纳滤波器会产生超过人眼所希望的严重的低通滤波效应，在同样条件下，γ<1

的参数化维纳滤波和几何均值似乎能给出更令人满意的结果。

[例8.3]维纳滤波器

本例采用基本滤波方法，输入正弦信号，并添加高斯噪声，通过求概率密度与功率谱曲线，对比普通的fir滤波和维纳滤波效果，程序如下所示。

```
%输入信号
A=1;    %信号的幅值
f=1000;    %信号的频率
fs=10^5;    %采样频率
t=(0:999);    %采样点
Mlag=100;    %相关函数长度变量
x=A*cos(2*pi*f*t/fs);    %输入正弦波信号
xmean=mean(x);    %正弦信号均值
xvar=var(x,1);    %正弦信号方差
xn=awgn(x,5);    %给正弦信号输入信噪比为20db的高斯白噪声
figure(1);
plot(t,xn)    %绘制输入信号图像
title('输入信号图像');
xlabel('x轴单位：t/s','color','b');
ylabel('y轴单位：f/Hz','color','b');
xnmean=mean(xn);    %计算添加噪声后的信号均值
xnvar=var(xn,1);    %方差
xnms=mean(xn.^2);    %均方值
Rxn=xcorr(xn,Mlag,'biased'); %自相关函数
figure(2)
subplot(221);
plot((-Mlag:Mlag),Rxn) %绘制自相关函数图
```

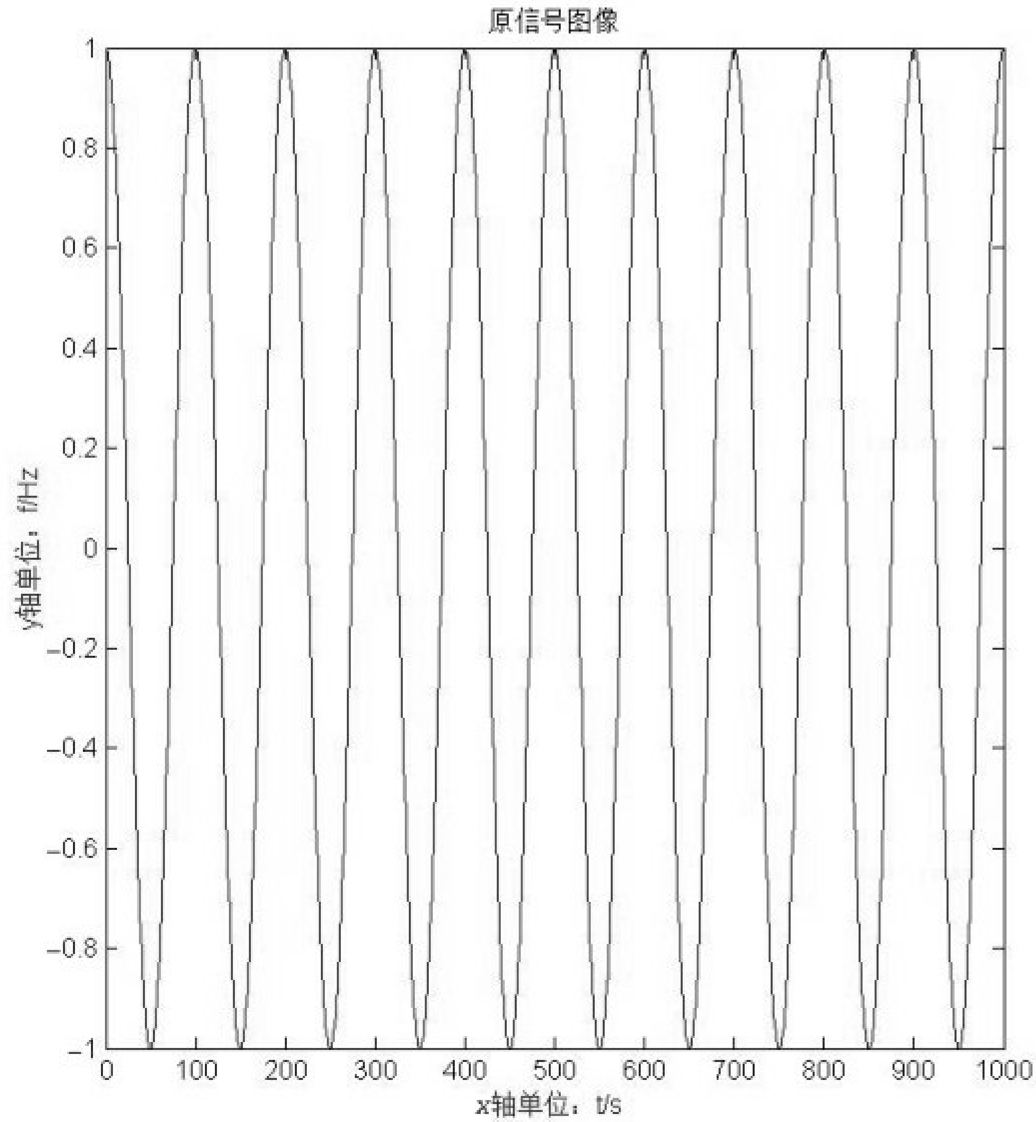
```
title('输入信号自相关函数图')
[f,xi]=ksdensity(xn); %计算输入信号的概率密度
subplot(222);
plot(xi,f);   %绘制概率密度图像
title('输入信号概率密度图像')
X=fft(xn);   %计算输入信号的快速傅立叶变换
Px=X.*conj(X)/600;   %计算信号频谱
subplot(223)
semilogy(t,Px)   %绘制在半对数坐标系下频谱图像
title('输入信号在半对数坐标系下频谱图像');
xlabel('x轴单位:w/rad','color','b');
ylabel('y轴单位:w/Hz','color','b');
pxx=periodogram(xn); %计算输入信号的功率谱密度
subplot(224);
semilogy(pxx);   %绘制在半对数坐标系下功率谱密度图像
title('输入信号在半对数坐标系下功率谱密度图像');
xlabel('x轴单位:w/rad','color','b');
ylabel('y轴单位:w/Hz','color','b');
%fir滤波
wp=0.4*pi;   %通带截止频率
ws=0.6*pi;   %阻带截止频率
DB=ws*wp;   %过渡带宽度
N0=ceil(6.6*pi/DB);
M=N0+mod(N0+1,2); %计算fir滤波器阶数
wc=(wp+ws)/2/pi; %计算理想低通滤波器通带截止频率
hn=fir1(M,wc);
y1n=filter(hn,1,xn); %将输入信号通过fir滤波器
```

```
figure(3)
plot(y1n);
title('经过fir滤波器后信号图像');
xlabel('x轴单位:f/Hz','color','b');
ylabel('y轴单位:A/V','color','b');
y1nmean=mean(y1n);
y1nms=mean(y1n.^2);
y1nvar=var(y1n,1);
Ry1n=xcorr(y1n,Mlag,'biased');
figure(4);
subplot(221);
plot((-Mlag:Mlag),Ry1n)
title('经过fir滤波器后信号自相关函数图像');
[f,y1i]=ksdensity(y1n);
subplot(222);
plot(y1i,f);
title('经过fir滤波器后信号概率密度图像');
Y1=fft(y1n);
Py1=Y1.*conj(Y1)/600;
subplot(223);
semilogy(t,Py1);
title('经过fir滤波器后信号在半对数坐标系下频谱图像');
xlabel('x轴单位:w/rad','color','b');
ylabel('y轴单位:w/Hz','color','b');
py1n=periodogram(y1n);
subplot(224);
semilogy(py1n);
```

```
title('经过fir滤波器后信号在半对数坐标系下功率谱密度图像');
xlabel('x轴单位:w/rad','color','b');
ylabel('y轴单位:w/Hz','color','b');
%维纳滤波
N=100;   %维纳滤波器长度
Rxnx=xcorr(xn,x,Mlag,'biased'); %产生输入信号与原始信号的互相关函数
rxnx=zeros(N,1);
rxnx(:)=Rxnx(101:101+N-1);
Rxx=zeros(N,N);
Rxx=diag(Rxn(101)*ones(1,N));
for i=2:N
  c=Rxn(101+i)*ones(1,N+1-i);
  Rxx=Rxx+diag(c,i-1)+diag(c,-i+1);
end
Rxx;
h=zeros(N,1);
h=inv(Rxx)*rxnx;
yn=filter(h,1,xn);
figure(5);
plot(yn)
title('经过维纳滤波器后信号图像');
xlabel('x轴单位:f/Hz','color','b');
ylabel('y轴单位:A/V','color','b');
ynmean=mean(yn);
ynms=mean(yn.^2);
ynvar=var(yn,1);
```

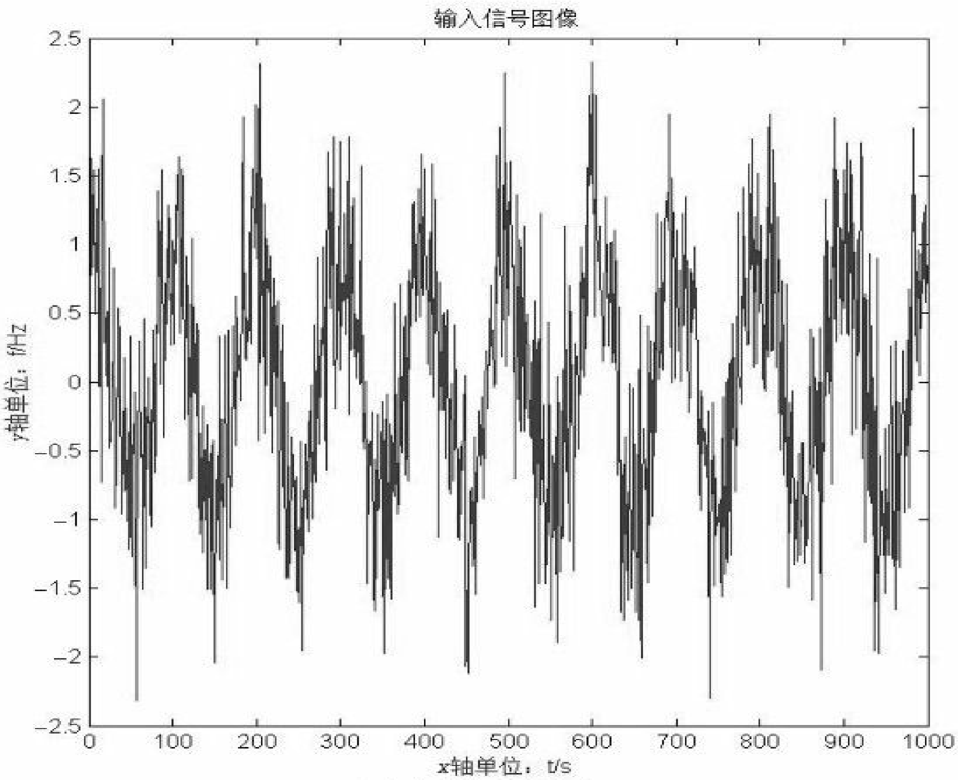
```
figure(6);
subplot(221);
plot((-Mlag:Mlag),Rxnx);
title('经过维纳滤波器后信号自相关函数图像');
[f,yi]=ksdensity(yn);
subplot(222);
plot(yi,f);
title('经过维纳滤波器后信号概率密度图像');
y=fft(yn);
Py=Y.*conj(Y)/600;
subplot(223);
semilogy(t,Py);
title('经过维纳滤波器后信号在半对数坐标系下频谱图像');
xlabel('x轴单位:w/rad','color','b');
ylabel('y轴单位:w/Hz','color','b');
pyn=periodogram(yn);
subplot(224);
semilogy(pyn);
title('经过fir滤波器后信号在半对数坐标系下功率谱密度图像');
xlabel('x轴单位:w/rad','color','b');
ylabel('y轴单位:w/Hz','color','b');
```

运行结果如图8.4所示，从图8.4中可以看出对于该信号图像添加相应的噪声信号后，从自相关函数、概率密度函数图像、频谱、功率谱等多个角度进行了对比，结果表明使用维纳滤波的效果要比使用fir滤波的效果好一些，经过fir滤波的信号去噪效果不太明显。

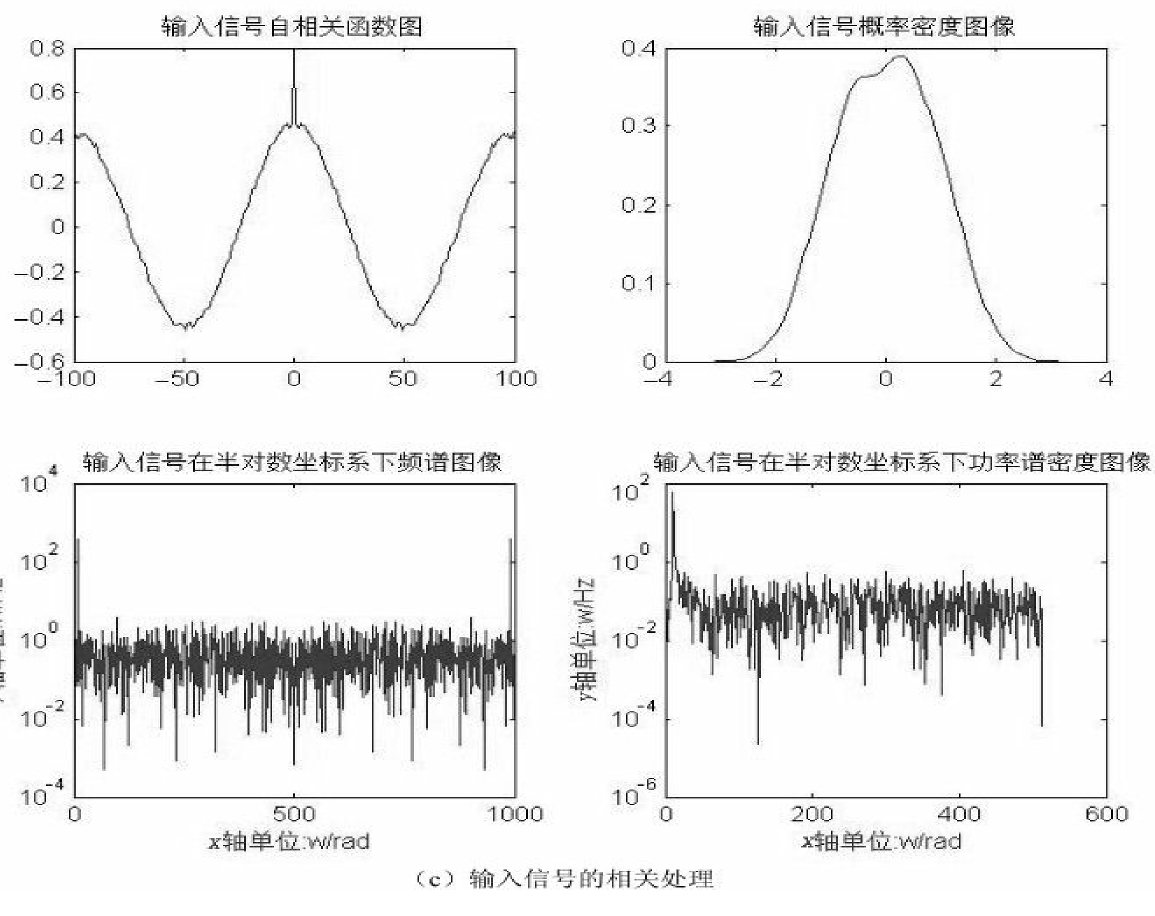


（a）原始信号

▲图8.4 运行结果

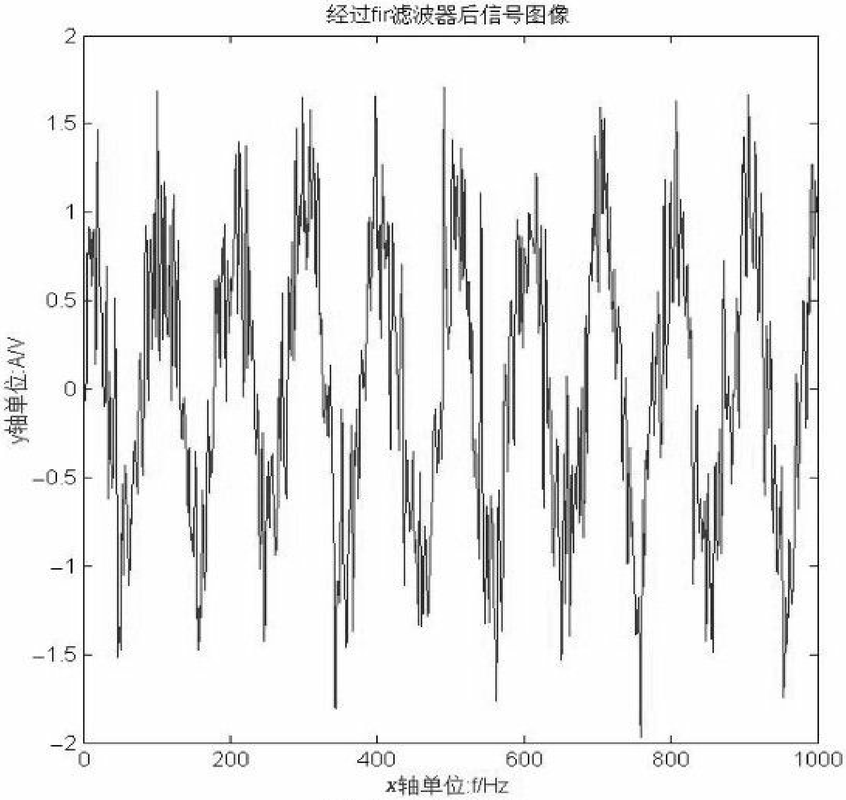


（b）加噪声后的输入

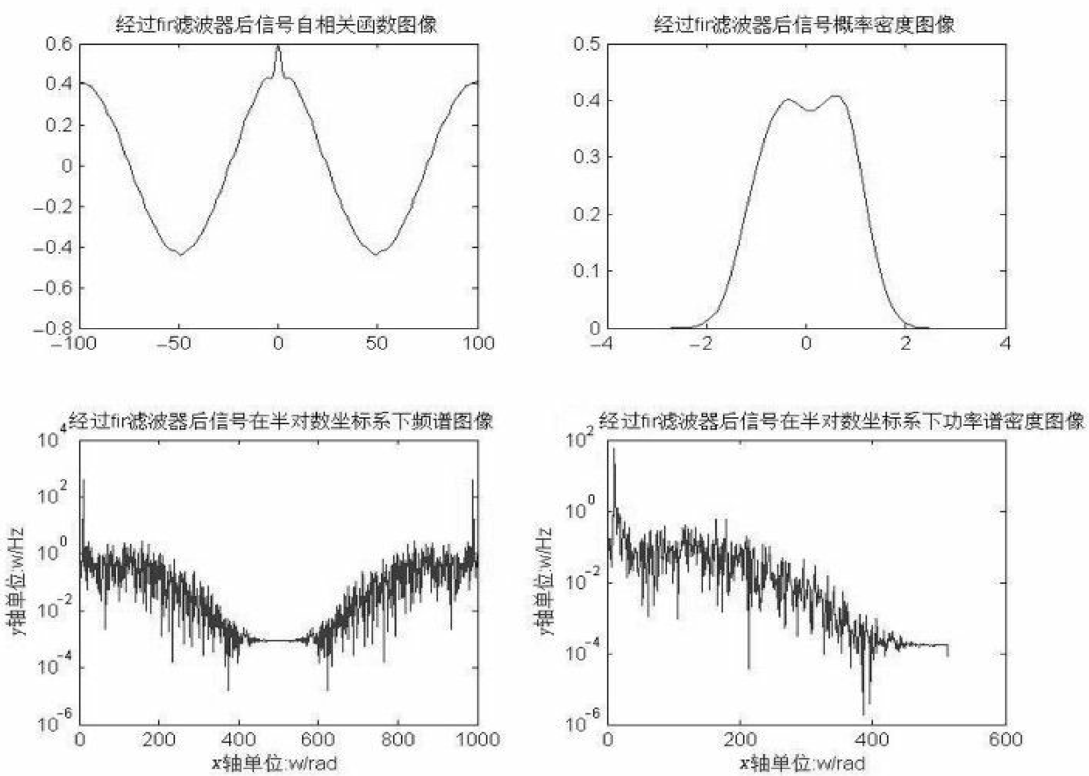


（c）输入信号的相关处理

▲图8.4 运行结果（续）

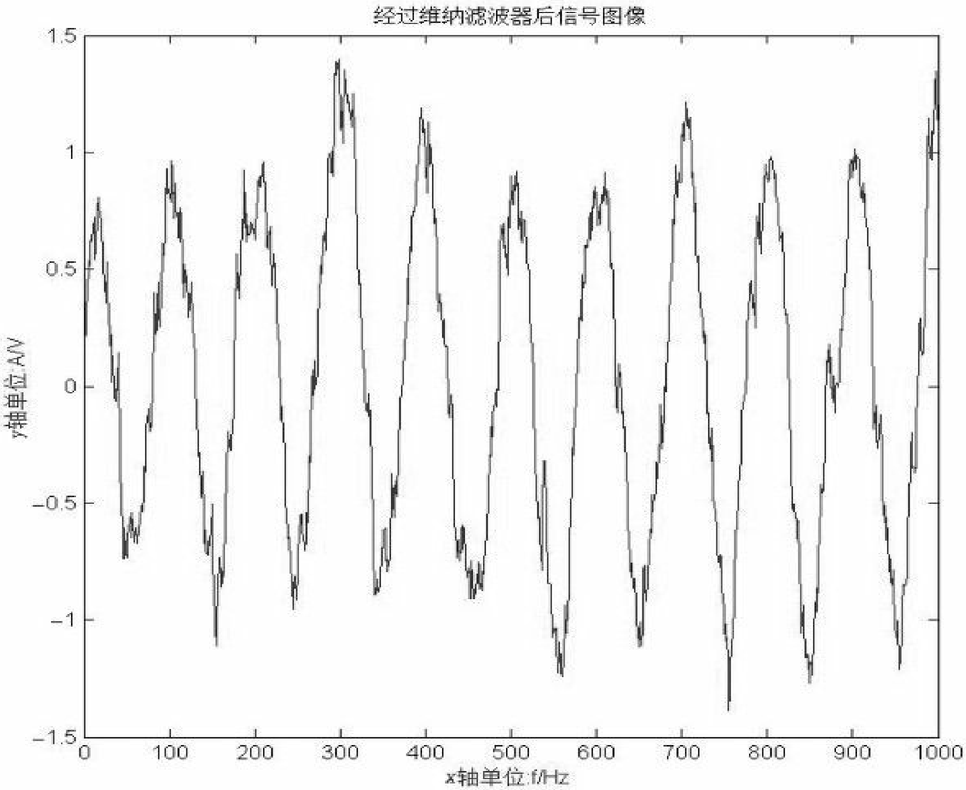


（d）fir 滤波后图像

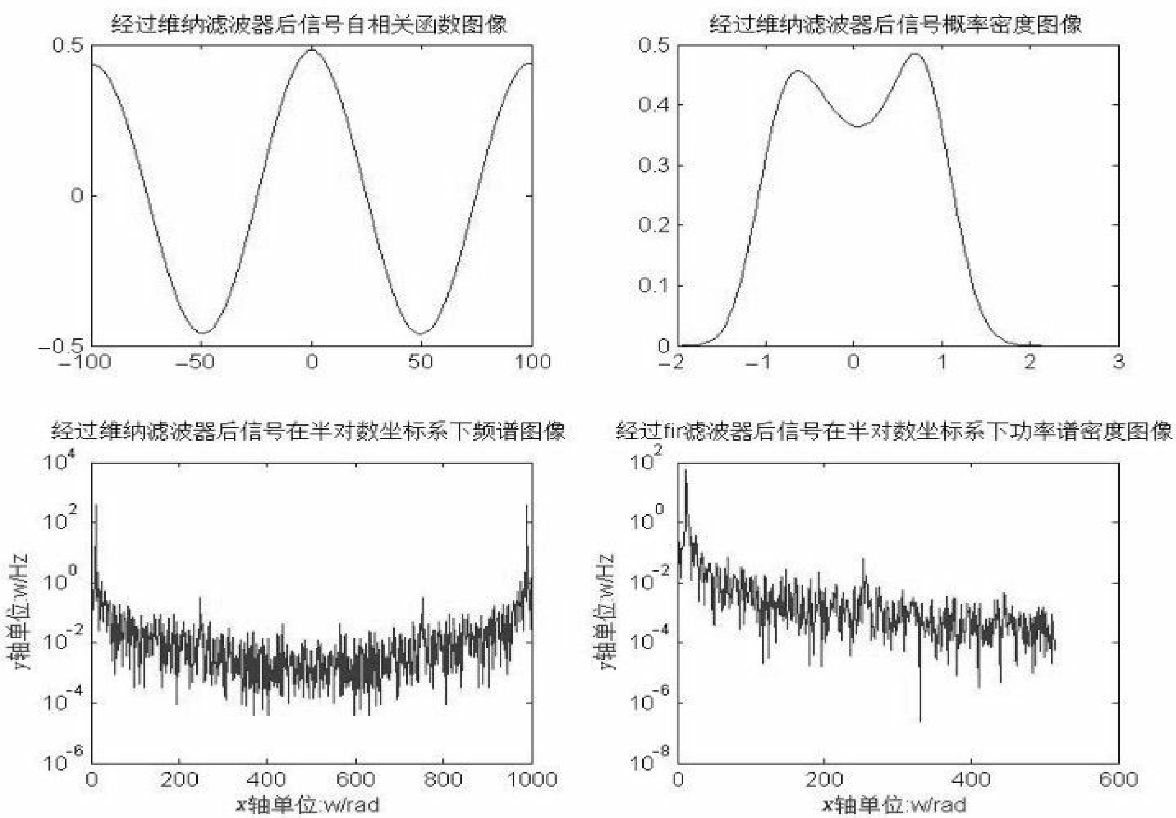


（e）对滤波后图像进行处理

▲图8.4 运行结果（续）



（f）维纳滤波后图像



（g）维纳滤波后的图像处理

▲图8.4 运行结果（续）

8.2 实用图像复原技术

图像复原技术早期的成果主要归功于数字信号处理领域中一些技术和方法的引入，如逆滤波技术。随后一些学者发现退化问题可以利用状态空间、非线性参数辨识、自适应理论等方法很好地描述和解决，这些方法具有典型的现代控制技术的特点。控制技术在复原技术领域的成功应用推进了复原技术的发展。在此之前，图像复原所依据的数值计算方法也是实现图像复原技术的关键。本节主要介绍数值计算方法与图像复原中涉及到的几何变换。

8.2.1 图像复原的数值计算方法

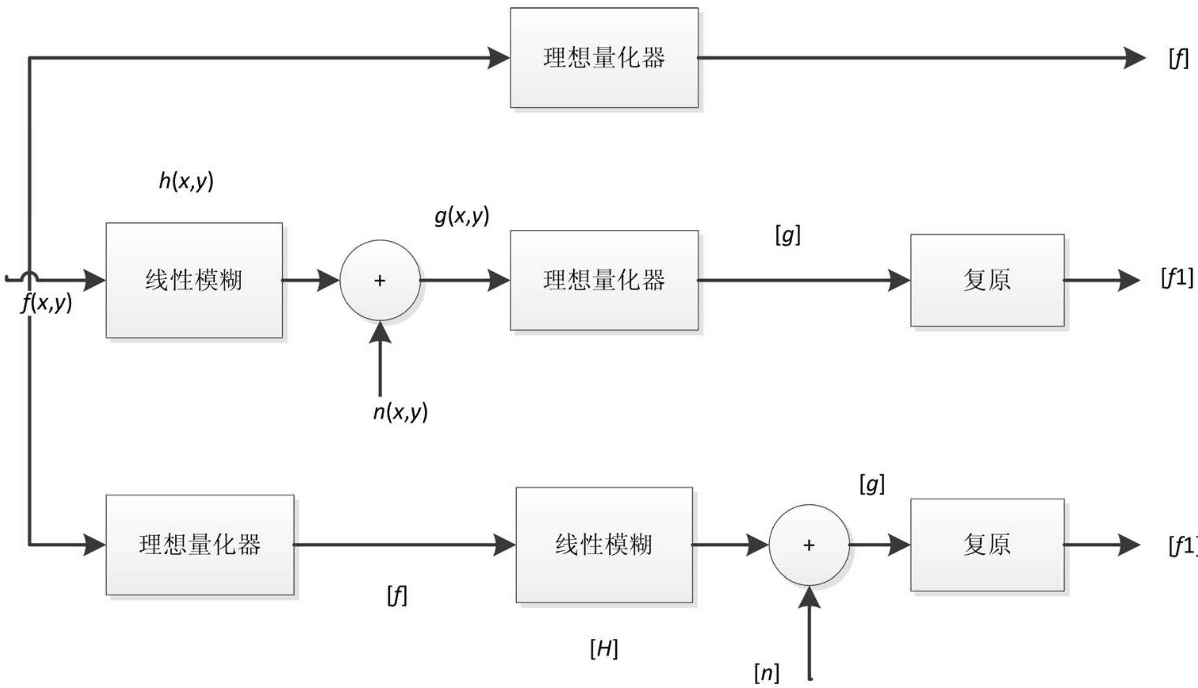
图像复原过程中，数值计算是不可避免的。下面介绍几种常用的数值计算方法：线性代数复原、无约束复原、有约束复原，并介绍几个相关例子来说明它们的计算方法。

1．线性代数复原

如果图像退化系统为线性空间不变系统，且噪声是加性类型时，可在线性代数范畴内建造的图像退化模型的基础上，采用代数方法进行数值计算。这种方法可能比较适合那些相对于积分运算而更喜欢矩阵运算，或相对于分析连续函数而更喜欢离散数学的人。它为复原滤波器的数值计算提供了一个统一的设计思路和透彻的解释。由于涉及的向量和矩阵都非常复杂，因此线性代数方法可能无法给出一种高效的实现。但依据线性代数方法得出的复原技术可通过其他方法得以高效的实现。

图8.5给出了在研究离散复原问题时要用到的模型。其最上一行表示了我们希望的情形，即用一个理想的量化器对代表景物未退化图像的

连续函数 f(x,y)进行操作以得到数字化图像。这将产生一个行堆叠形式的$n^2\times1$列向量f，代表着所需的N×N数字图像。



▲图8.5 线性代数复原模型

图 8.5 中第二行给出一幅图像被量化和存储时，实际所发生情况的模型。景物函数 f(x,y)被一个线性操作h(x,y)所模糊，然后叠加上一幅二维噪声图像n(x,y)，得到g(x,y),随后由一个理想量化器生成行堆叠形式的向量g，它代表被观察的N×N数字图像；然后对这幅图像进行复原操作，以产生对原图像f的估计f1。

虽然模糊是线性的，但它可能是位移不变的。如果是位移不变的，它就等于 f(x,y)与点扩散函数 h(x,y)的卷积。如果成像链中包含不止一个操作环节，它们的影响统一通过 h(x,y)来体现。与此类似，多个噪声源也用一个n(x,y)来表示。由于此模型未考虑各种非线性以及信号相关的噪声，因此它仍然不够全面。

图中的第三行给出了在这里将要分析的模型。如前，一个理想的量

化器生成向量f，但它受到离散线性操作H的作用。同时，一幅离散的、堆叠成为列向量形式的噪声图像n被加到f上形成列向量形式的被观察图像g；再通过一个离散的复原操作来得到原始图像的估计。

至此，被观察图像的形成过程可以简洁的表示为

$$g = Hf + n \qquad (8\text{-}28)$$

式中：g、f和n都是n^2维列向量；H为$n^2 \times n^2$的矩阵。若模糊是位移不变的，则H为块循环矩阵。同前，要处理的数字图像应按需要用0填充到N×N大小。

要注意的是，现在是在用离散操作来模拟图像的退化，它包括两个方面：其一，既然可以对退化过程进行设计并加以精确实现，应该就能利用这个模型产生给人深刻印象的仿真结果。例如，若选择一个可逆的退化过程，复原就成了仅仅是数字技术的一个练习。可以首先对它进行退化，再进行复原，并得到精确到舍入误差的范围之内的原始图像。

其次，现在的任务是用离散操作模拟连续退化过程。这与早先要求对采样数据的离散处理必须保证确实能保留其本来连续函数的完整性这一情况相类似。图像复原的有效性关键取决于描述退化过程的模型的精确性。

2．无约束复原

若n=0或者对于噪声一无所知，则可用如下方法把复原问题当作一个最小二乘问题来解决。令e(f1)为f1与其近似向量f之间的残差向量，则式（8-28）变为

$$g = Hf = Hf1 + e(f) \text{或} e(f1) = g - Hf1 \qquad (8\text{-}29)$$

我们的目标是使目标函数

$$W(f1) = \| ef1 \|^2 = \| g - Hf1 \|^2 = (g - Hf1)^t (g - Hf1) \qquad (8\text{-}30)$$

最小化，其中，$\| a \| = \sqrt{a^t a}$ 代表一个向量的欧几里得范数，即其各元素平方和的平方根。

这意味着我们希望这样来选择f1，使它被H模糊后所得到的结果与观察到的图像g之间的差的均方尽可能小。由于g本身就是由f经H模糊得到的，可以想象，这是一种令人满意的方法。若f和f1这两者被H模糊的结果接近相等，则f1很可能就是对f的一个良好估计。

令W(f1)对f1的导数等于零，得到

$$\frac{\partial W(f1)}{\partial f1} = -2H^t(g - Hf1) = 0 \qquad (8\text{-}31)$$

求解f1，得到

$$f1 = (H^t H)^{-1} H^t g = H^{-1} g \qquad (8\text{-}32)$$

由于H为方阵，故后一等式成立。

式（8-32）给出的就是逆滤波器。对于位移不变的模糊，H为块循环矩阵。这可以通过在频率域用下式进行卷积去说明。

$$\hat{F}(u,v) = \frac{G(u,v)}{H(u,v)} \qquad (8\text{-}33)$$

若H(u,v)有零值，则H为奇异的，无论H^{-1}或（$H^t H)^{-1}$都不存在。

3．有约束最小二乘复原

重写式（8-28）为

$$g - Hf = n \qquad (8\text{-}34)$$

考虑噪声项的一种方法是在极小化过程中引入要求式（8-27）两端范数相等的约束，即

$$\| g - Hf1 \| = \| n \|^2 \qquad (8\text{-}35)$$

现在，可以将问题归结为最小化如下目标函数。

$$W(f1) = \| Qf1 \|^2 + \lambda(\| g - Hf1 \|^2 - \| n \|^2) \qquad (8\text{-}36)$$

式中：Q 是选来对 f1 进行某性运算的矩阵；λ为一常数，叫做拉格朗日因数。通过指定不同的Q，可以达到不同的复原目标。

如前，令W(f1)对f1的导数为零，有

$$\frac{\partial W(f1)}{\partial f1} = 2Q^t Qf1 - 2\lambda H^t(g - Hf1) = 0 \qquad (8\text{-}37)$$

解得

$$f1 = (H^t H + \gamma Q^t Q)^{-1} H^t g \qquad (8\text{-}38)$$

式中：γ=1/λ为一必须经过调整，以使式（8-35）成立的常数。这是求有约束最小二乘复原的解得通用方程式。

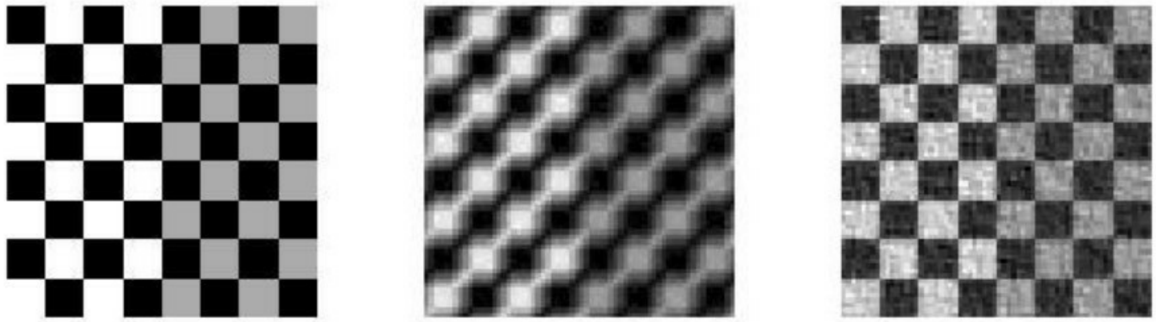
[例8.4]利用约束的最小二乘方滤波复原模糊噪声图像

约束的最小二乘方滤波在Matlab中是通过函数deconvreg来实现的。该例说明了如何利用该函数复原模糊噪声图像。程序如下。

```
%生成标准棋盘测试图像
f=checkerboard(8);
%产生空间滤波器
PSF=fspecial('motion',7,45);
>> PSF
PSF =
     0         0         0         0         0    0.0145         0
     0         0         0         0    0.0376    0.1283    0.0145
```

```
        0         0         0    0.0376    0.1283    0.0376         0
        0         0    0.0376    0.1283    0.0376         0         0
        0    0.0376    0.1283    0.0376         0         0         0
   0.0145    0.1283    0.0376         0         0         0         0
        0    0.0145         0         0         0         0         0
gb=imfilter(f,PSF,'circular');
%产生噪声模式
noise=imnoise(zeros(size(f)),'gaussian',0,0.001);
g=gb+noise;
subplot(1,3,1);
%将原图像放大到512*512，并显示
imshow(pixeldup(f,8),[]);
subplot(1,3,2);
%将噪声图像放大到512*512，并显示
imshow(pixeldup(g,8),[]);
%利用函数deconvreg()复原模糊噪声图像
fr=deconvreg(g,PSF,0.4,[1e-7 1e7]);
subplot(1,3,3);
imshow(pixeldup(fr,8),[])
```

运行结果如图8.6所示，对比噪声模糊图8.6（b）和滤波后的图8.6（c），可以看出，有约束的最小二乘方滤波一定程度上能改善原模糊噪声图像的质量。



（a）产生的测试板图像　（b）使用噪声污染的图像　（c）使用约束最小二乘复原的图像

▲图8.6

8.2.2 非线性复原

现实例子中，除了前面讨论的线性空间不变系统，还有许多不受这些条件限制的系统，本节主要讲述这些系统的图像非线性复原方法。

1．随空间改变的系统

如果说光学离焦与线性运动模糊是具有空间不变性的线性操作，那么像散、彗差、像场弯曲以及旋转运动等模型则是随空间改变的。纠正这些退化的一种直接而有效的复原方法是坐标变换复原。其思想是通过对退化图像进行几何变换，使得到的函数具有空间不变性。随后采用普通的空间不变复原方法对其进行复原，再用一个和先前几何变换相反的逆变换将图像复原为原来的格式。

2．时变模糊

200英寸望远镜的衍射极限分辨率为0.05弧秒。然而在恶劣条件下，大气扰动会使分辨率降低到2弧秒。通过扰动的大气层观察星星，就仿佛是通过一扇运动着的浴室门上的花玻璃观察一个点光源一样。

在曝光时间短的情况下，由于望远镜上非均匀的大气层造成的相位扭曲，将会产生一个斑点图案。在曝光时间较长的情况下，由于大气层在变化，其扰动将使斑点图案游动。因此，长时间的曝光使这些游动的光斑累积成一个远远超过望远镜的衍射极限点扩散函数的相当大的模

糊。由于在拍摄暗淡星体的照片时必须长时间曝光，大气扰动限制了地面观测的天文分辨率。

在空间域进行时间平均，等效于在频域对复频谱的平均。如此得到的时间平均传递函数在远低于望远镜衍射极限的频率处就下降到零，因此在有随机相位畸变的情况下，时间平均带来的害处比好处多。

Labeyrie通过实验证明，点星体图像的时间平均功率谱一直分布到衍射极限以外。这意味着大气层的相位随机波动在图像的功率谱中被平均掉了。他采用了如下的复原技术。首先获得欲观测的天体和一个参照的点星体两者的时间平均功率谱；然后通过将该天体的功率谱除以点星体的功率谱实现去卷积，所得结果就是对未知天体的衍射极限功率谱的估计，将其逆变换就得到该天体的自相关函数。由于相位信息在功率谱中被丢失了，天体无法准确地重建，但用自相关函数足以认清双星和其他可能感兴趣的天体。

Knox将Labeyrie的方法进一步推广，使其复原相位信息，这样即使在相对恶劣的观察条件下也能得到衍射极限图像。与Labeyrie类似，他用短时曝光功率谱的整体平均来确定目标的功率谱，相位信息是通过即时功率谱的整体自相关来获得的。

3．非平稳信号与噪声

本节前面讨论的滤波器均假设信号和噪声是平稳的。平稳对于图像来说意味着局部（算出的）功率谱在整幅图像上是相同或近似相同的。不过，通常并非如此。实际上，大多数图像是高度不平稳的。以一张人脸的照片为例，显然前额区的功率谱比眼睛处的功率谱要含有较少的高频分量。很大的一类图像可被认为是由被梯度相对高的边缘所隔开的、相对平坦的区域所组成。

有几种常见的噪声也不能用平稳随机过程来精确建模。比如底片的颗粒噪声，在底片的低密度区几乎不存在，但随着密度的增加噪声也相应增强。密度量化器，由于其在强度传感器后接有对数放大器，而对数

放大器的弱信号增益是最大的，因此在暗区域就会产生相对比较高的噪声。

4．使用**Lucky-Richardson**算法的迭代非线性复原

Lucky-Richardson（L-R）算法是非线性方法中一种典型的算法。L-R算法是从最大似然公式中引出来的，在这个方程中，图像是用泊松统计加以模型化的。当下面这个迭代收敛时，模型的最大似然函数可以得到一个令人满意的方程，即

$$\hat{f}_{k+1}(x,y)=\hat{f}_{k}(x,y)[h(-x,-y)*\frac{g(x,y)}{h(x,y)*\hat{f}_{k}(x,y)}] \tag{8-39}$$

如前边一样，“*”代表卷积，$\hat{f}$是未退化图像的估计。这个算法的迭代本质是显而易见的。它的非线性本质是在方程右边用$\hat{f}$来除产生的。

就像大多数非线性方法一样，关于什么时候停止L-R算法通常很难回答。通常，接下来的途径是对于给定的应用，在我们获得满意的结果时，观察输出并终止算法。

[例8.5]使用L-R算法复原模糊噪声图像

对于前面提到的系统，一般采用迭代非线性复原的方法。在Matlab工具箱中选择的算法是L-R算法。该算法是由名为deconvlucy的函数完成的。程序如下。

```
%产生标准测试图像，64*64像素
f=checkerboard(8);
subplot(3,2,1);
%放大显示
imshow(pixeldup(f,8));
%产生大小为7*7且标准偏差为10的高斯滤波器
```

PSF=fspecial('gaussian',7,10);

%用上述滤波器模糊图像f，在其上添加均值为0，标准偏差为0.01的高斯噪声

SD=0.01;

g=imnoise(imfilter(f,PSF),'gaussian',0,SD^2);

subplot(3,2,2);

imshow(pixeldup(g,8));

%使用Lucy-Richardson算法对图像复原

DAMPAR=10*SD;

LIM=ceil(size(PSF,1)/2);

WEIGHT=zeros(size(g));

%weight数组的大小是64*64，并且有值为0的4像素宽的边界，其余像素都是1

WEIGHT(LIM+1:end-LIM,LIM+1:end-LIM)=1;

%设置迭代次数为5

NUMIT=5;

%利用deconvlucy来实现复原

f5=deconvlucy(g,PSF,NUMIT,DAMPAR,WEIGHT);

subplot(3,2,3);

imshow(pixeldup(f5,8));

%设置迭代次数为10

NUMIT=10;

f6=deconvlucy(g,PSF,NUMIT,DAMPAR,WEIGHT);

subplot(3,2,4)

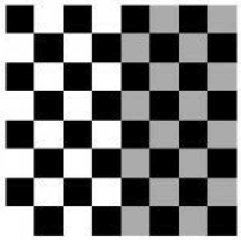
imshow(pixeldup(f6,8));

%设置迭代次数为20

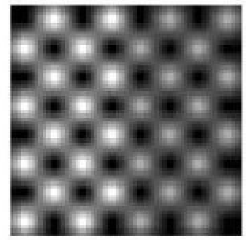
NUMIT=20;

```
f7=deconvlucy(g,PSF,NUMIT,DAMPAR,WEIGHT);
subplot(3,2,5);
imshow(pixeldup(f7,8));
%设置迭代次数为100
NUMIT=100;
f8=deconvlucy(g,PSF,NUMIT,DAMPAR,WEIGHT);
subplot(3,2,6);
imshow(pixeldup(f8,8));
```

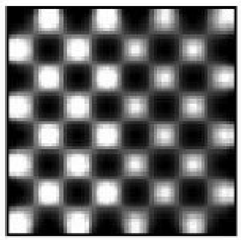
运行结果如图8.7所示，图8.7（c）是经过5次迭代后的结果，虽然图像已经稍微有些改进了，但是仍旧模糊。图8.7（d）和图8.7（e）是迭代10次和20次的结果，进一步增加迭代次数在复原结果上没有显著改进，图8.7（f）是迭代100次的结果，这幅图像只是比使用了20次迭代后获得的图像稍微清晰和明亮了一些。



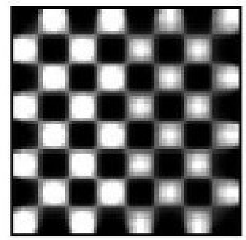
（a）原图像



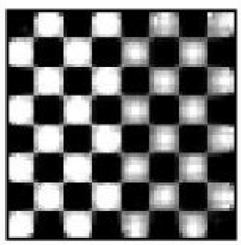
（b）由高斯噪声污染和模糊的图像



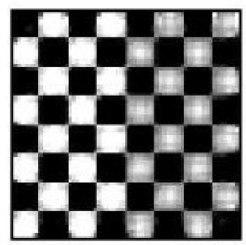
（c）对图像（b）用 L-R 算法迭代 5 次后的复原图像



（d）对图像（b）用 L-R 算法迭代 10 次后的复原图像



（e）对图像（b）用 L-R 算法迭代 20 次后的复原图像



（f）对图像（b）用 L-R 算法迭代 100 次后的复原图像

▲图8.7 运行结果

第9章 彩色图像处理

随着基于互联网的图像处理应用在不断增长，彩色图像处理已经成为一个重要领域。本章以介绍色彩模型为主，并将涉及数字域彩色处理方面的基本概念与常识。

本章的知识和技术热点

- 彩色基础
- 彩色模型及其相互转换
- 全彩色图像处理

本章的典型案例分析

- 彩色补偿与色彩平衡

9.1 彩色基础

大千世界，光彩绚丽，色彩让世界变得更加迷人。那么，色彩是什么呢？色彩又有什么特性呢？在17世纪60年代，人们还普遍认为白光是一种纯的没有其他颜色的光，而彩色光是一种不知何故发生变化的光。为验证该假设，牛顿让一束太阳光透过一面三棱镜，光在墙上被分解为7种不同的颜色，即红、橙、黄、绿、青、蓝、紫，后来我们称之为光谱，如图9.1所示。



▲图9.1 牛顿发现彩色谱现象

正是由于这些红、橙、黄、绿、青、蓝、紫基础色有不同的色谱，才形成了表面上颜色单一的白色光。虽然人的大脑感知和理解颜色所遵循的生理心理过程还远没有了解透彻，但颜色的物理性质得到了许多实验和理论结果的支持。

9.1.1 什么是彩色

彩色是物体的一种属性，就像纹理、形状、重量一样。通常，它依赖于3个方面的因素。

- 光源——照射光的谱性质或谱能量分布。
- 物体——被照射物体的反射性质。
- 成像接收器（眼睛或成像传感器）——光谱能量吸收性质。

其中，光特性是颜色科学的核心。假如光是没有颜色的（消色的，如观察者看到的黑白电视的光），那么它的属性仅仅是亮度或者数值。可以用灰度值来描述亮度，它的范围从黑到灰，最后到白。

而对于彩色光，我们通常用3个基本量来描述其光源的质量：辐射率、光强和亮度。

（1）辐射率是从光源流出能量的总量，通常用瓦特（W）度量。

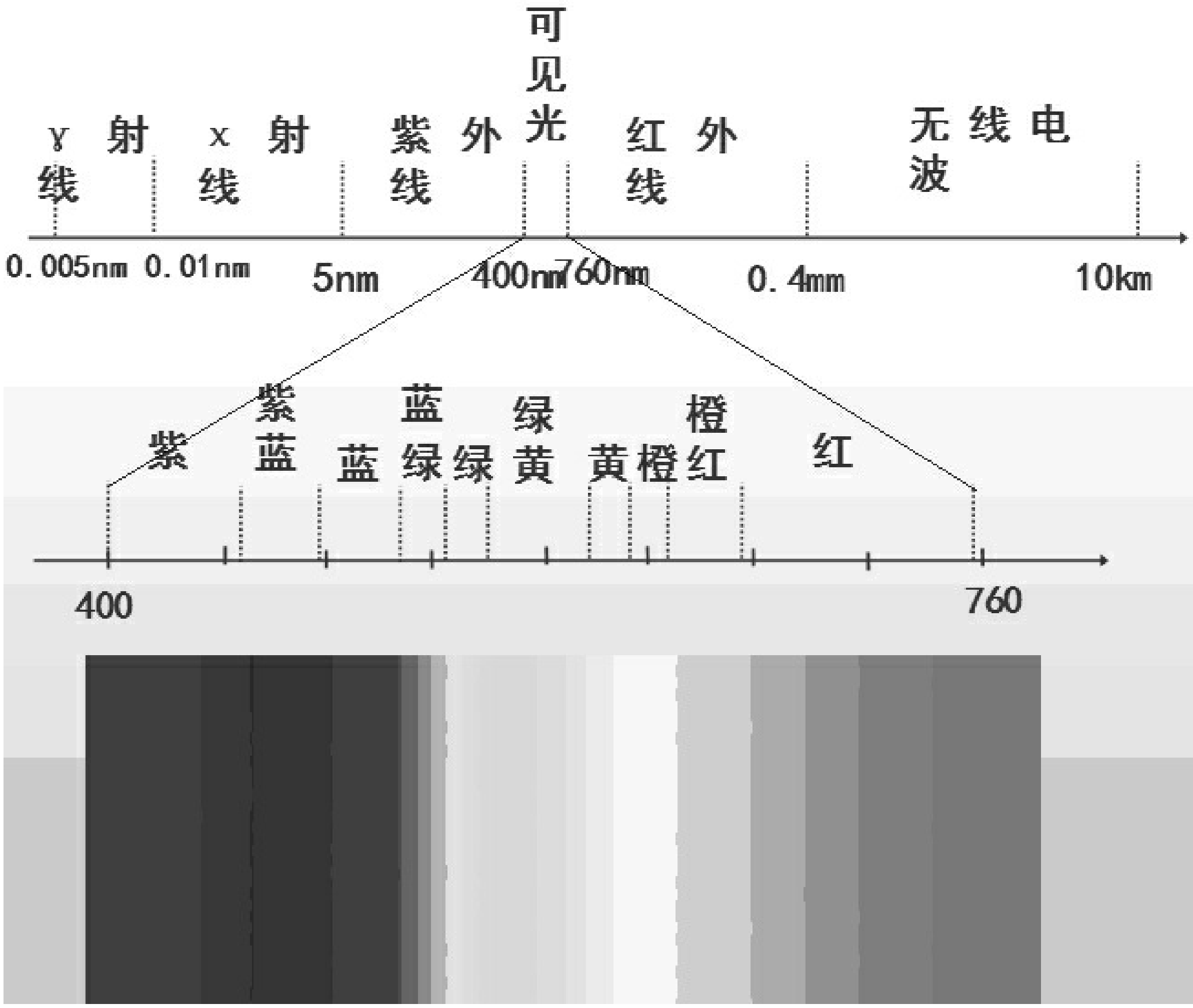
（2）光强用流明度量，它给出了观察者从光源接收的能量总和的度量。

（3）亮度是彩色强度概念的具体化。它实际上是一个难以度量的主关描绘。

同样作为能量的度量，辐射率与光强却往往没有必然的联系。例如，在进行X光检查时，光从X射线源中发出，它是具有实际意义上的能量的。但由于其处于可见光范围以外，作为观察者很难感觉到。因而对我们来说，它的光强几乎为0。

9.1.2 我们眼中的彩色

人类能够感受到的物体的颜色是由物体的反射光性质决定的。如图9.2所示，可见光是由电磁波谱中较窄的波段组成的。一个物体反射的光如果在所有可见光波长范围内是平衡的，则站在观察者的角度它就是白色的；如果物体仅对有限的可见光谱范围反射，则物体表现为某种特定颜色。例如，反射波长范围在450～500nm之间的物体呈现蓝色，它吸收了其他波长光的多数能量；而如果物体吸收了所有的入射光，则它将呈现为黑色。

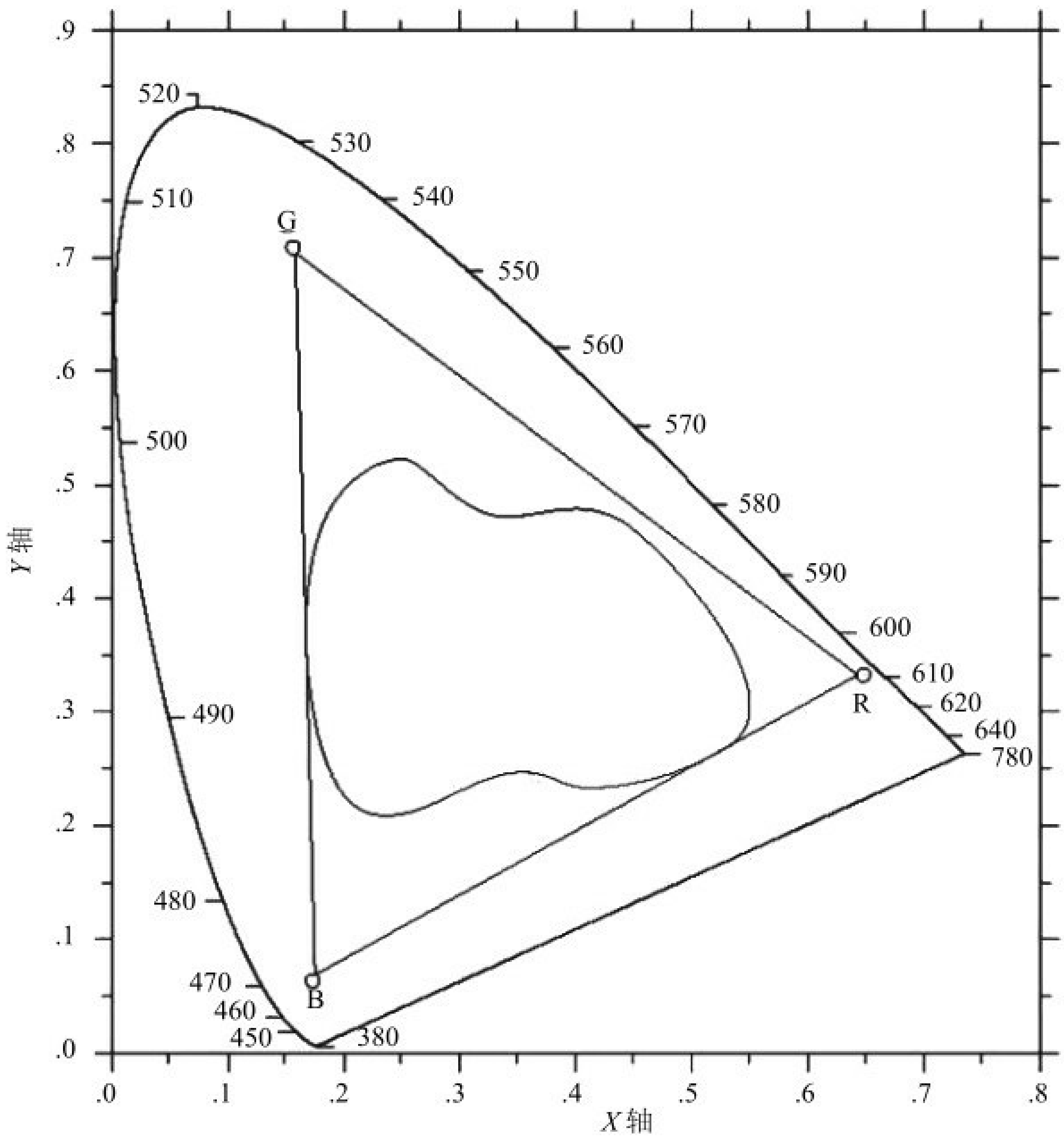


▲图9.2 可见光波长范围

9.1.3 三原色

据详细的实验结果，人眼中负责彩色感知的细胞中约有65%对红光敏感，33%对绿光敏感，而只有2%对蓝光敏感。正是人眼的这些吸收特性决定了被看到的彩色是通常所谓的原色红（R）、绿（G）、蓝（B）的各种组合。国际照明委员会（CIE）规定以蓝=435.8nm、绿=546.1nm、红=700nm作为主原色，红（R）、绿（G）、蓝（B）也因此被称为三原色。

在图9.3所示的CIE色度图中，最外围的轮廓对应着所有的可见光谱色，在其边缘上标出了对应的波长值（以 nm 为单位），该轮廓之内的区域包含了所有的可见颜色。如果将色度图中的三色点两两连接成一个三角形，则该三角形内的任何颜色都可以由这3种原色的不同混合产生。



▲图9.3 CIE色度图

我们看到，图 9.3中由R、G、B 3种标准原色所连成的三角形并不能涵盖整个可见颜色区域，这说明仅使用三原色并不能得到所有的可见颜色。事实上，图 9.3 中的三角形区域对应着典型的RGB 监视器所能够产生的颜色范围，称为彩色全域；而在三角形内不规则的区域表示高质量的彩色打印设备的彩色域。

9.1.4 计算机中的颜色表示

在计算机中，显示器的任何颜色（色彩全域）都可以由3种颜色红、绿、蓝组成，称为三基色。每种基色的取值范围是 0～255。任何颜色都可以用这 3 种颜色按不同的比例混合而成，这就是三原色原理。在计算机中，三原色的原理可以这样解释。

● 计算机中的任何颜色都可以由3种颜色按不同的比例混合而成；而每种颜色都可以分解成3种基本颜色。

● 三原色之间是相互独立的，任何一种颜色都不能由其余的两种颜色来组成。

● 混合色的饱和度由3种颜色的比例来决定。混合色的亮度为3种颜色的亮度之和。

形成任何特殊颜色需要的红、绿、蓝的分量称为三色值，我们可以用X、Y和Z分别表示。此时，一种颜色可由三色值系数定义为

$$x = \frac{X}{X+Y+Z}$$

$$y = \frac{Y}{X+Y+Z}$$

$$z = \frac{Z}{X+Y+Z}$$

显然：

$$x + y + z = 1$$

9.2 彩色模型

彩色模型也称彩色空间或彩色系统，是用来精确标定和生成各种颜色的一套规则和定义，它的用途是在某些标准下用通常可接受的方式简化彩色规范。彩色模型通常可以采用坐标系统来描述，而位于系统中的每种颜色都由坐标空间中的单个点来表示。

如今使用的大部分彩色模型都是面向应用的或是面向硬件的，比如众所周知的针对彩色监视器的 RGB（红、绿、蓝）模型，以及面向彩色打印机的 CMY（青、深红、黄）和 CMYK（青、深红、黄、黑）模型。而HSI（色调、饱和度、亮度）模型非常符合人眼描述和解释颜色的方式。此外，目前广泛使用的彩色模型还有HSV模型、YUV模型、YIQ模型、Lab模型等。下面将分别介绍这些彩色模型并给出它们与最为常用的RGB模型之间的转换方式。

9.2.1 RGB模型

RGB模型是工业界的一种颜色标准，是通过对红（Red）、绿

（Green）、蓝（Blue）3个颜色亮度的变化以及它们相互之间的叠加来得到各种各样的颜色的。该标准几乎包括了人类视觉所能感知的所有颜色，是目前运用最广的颜色模型之一。

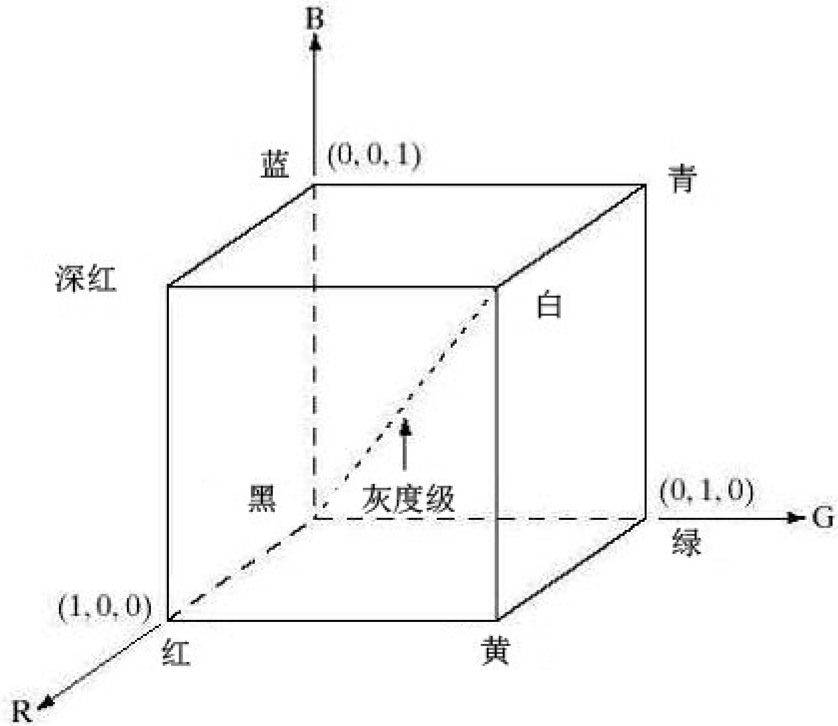
1．理论基础

RGB彩色空间对应的坐标系统是如图9.4所示的立方体。红、绿和蓝位于立方体的3个顶点上；青、深红和黄位于另外3个顶点上；黑色在原点处，而白色位于距离原点最远的顶点处，而灰度等级就沿这两点连线分布；不同的颜色处在立方体上或其内部，因此可以用1个三维向量来表示。例如，在所有颜色均已归一化至[0，1]的情况下，蓝色可表示为（0，0，1），而灰色可由向量（0.5，0.5，0.5）来表示。

在RGB模型中，3个图像分量组成了所要表示的图像，而每一个分量图像都是其原色图像，如图9.5所示。当送入RGB监视器时，这3个分量图像便在屏上混合产生一幅合成的彩色图像。

在RGB空间中，用以表示每一像素的比特数叫做像素深度。RGB图像的3个红、绿、蓝分量图像都是一幅8bit图像，则每一个彩色像素有24bit深度。因此，全彩色图像常用来定义24bit的彩色图像，颜色总数是$(2^8)^3$ =16 777 216。

RGB模型最常见的用途就是显示器系统，彩色阴极射线管和彩色光栅图形显示器都使用R、G、B数值来驱动R、G、B电子枪发射电子，并分别激发荧光屏上的R、G、B 3种颜色的荧光粉发出不同亮度的光线，并通过相加混合产生各种颜色。扫描仪也是通过吸收原稿经反射或透射而发送出来的光线中的R、G、B成分，并用它来表示原稿的颜色。RGB彩色空间是与设备相关的彩色空间，因此不同的扫描仪扫描同一幅图像，会得到不同色彩的图像数据；不同型号的显示器显示同一幅图像，也会有不同的色彩显示结果。



▲图9.4 RGB彩色立方示意图



（a）原图像 rgb.tif

（b）红色分量

（c）绿色分量

（d）蓝色分量

▲图9.5 RGB图像的 3个分量，注意图（a）中 RGB图像的不同颜色在其分量图中的对应强度。

注意

显示器和扫描仪使用的RGB空间与CIE标准中RGB真实三原色彩色系统空间是不同的，后者是与设备无关的彩色空间。

2．Matlab实现

在Matlab中一幅RGB图像可表示为一个M×N×3的三维矩阵。其中每一个彩色像素都在特定空间位置的彩色图像中对应红、绿、蓝3个分量。分量图像的数据类型决定了它们的取值范围。若一幅RGB图像的数据类型是double，则每个分量图像的取值范围为[0, 1]，而如果数据类型为uint8或 uint16，则每个分量图像的取值范围分别是[0, 255]或[0, 65 535]。

● 图像合成

如果令PR、PG、PB分别代表3种RGB分量。那么一幅RGB图像就是利用cat（级联）操作符将这些分量图像组合成彩色图像：

RGB_image=cat(3, PR, PG, PB); % 将PR、PG、PB3个矩阵在第3个维度上进行级联

注意在上述cat操作中，图像应按照R、G、B的顺序放置。如果所有的分量图像都相等，则结果将是一幅灰度图像。

● 分量提取

令RGB_image代表一幅RGB图像，下面的命令可以提取3个分量图像：

PR=RGB_image(:,:,1);

PG=RGB_image(:,:,2);

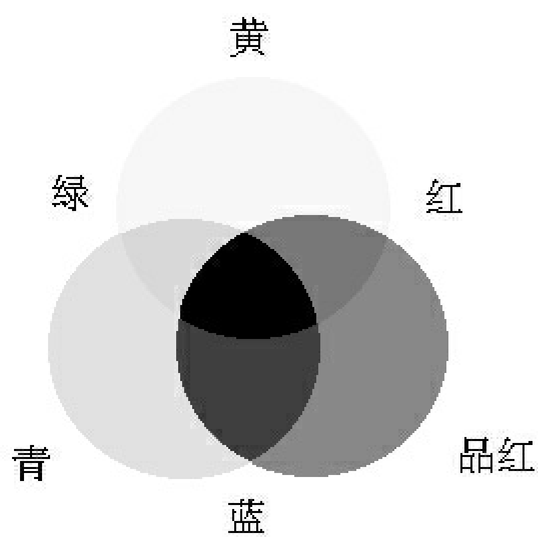
PB=RGB_image(:,:,3);

9.2.2 CMY、CMYK模型

1．CMY模型

CMY模型（Cyan、Magenta、Yellow）是采用青、品红、黄色3种基本原色按一定比例合成颜色的方法。由于色彩的显示不是直接来自于光线的色彩，而是光线被物体吸收掉一部分之后反射回来的剩余光线所产生的，因此CMY模型又称为减色法混色模型。当光线都被吸收时成为黑色，都被反射时成为白色。

像CMY模型这样的减色混合模型正好适用于彩色打印机和复印机这类需要在纸上沉积彩色颜料的设备，因为颜料不是像显示器那样发出颜色，而是反射颜色。例如，当青色颜料涂覆的表面用白光照射时，从该表面反射的不是红光，而是从反射的白光中减去红色而得到的青色（白光本身是等量的红、绿、蓝光的组合）。CMY 模型的颜料混合效果如图 9.6 所示，注意这里的混合是原色的相减，与RGB模型的混合正好相反。



▲图9.6 CMY模型的颜色混合（原色相减）

2．CMYK模型

由图9.6可见，等量的颜料原色（青、品红和黄）可以混合产生黑色。然而在实际当中，通过这些颜色混合产生的黑色是不纯的。因此，为产生真正的黑色（黑色在打印中起主要作用），专门在CMY模型中加入了第4种颜色——黑色，从而得到了CMYK彩色模型。这样当出版商说到“四色打印”时，是指CMY彩色模型的3种原色再加上黑色。

3．**RGB**与**CMY**之间的转换及其实现

RGB与CMY之间的转换公式如下。

$$\begin{bmatrix} C \\ M \\ Y \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} R \\ G \\ B \end{bmatrix} \qquad \begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} C \\ M \\ Y \end{bmatrix}$$

其中，假设所有的颜色值都已经归一化到范围[0,1]。

● Matlab实现

在Matlab中可以通过imcomplement函数方便地实现RGB和CMY之间的相互转换。

```
cmy = imcomplement (rgb);
rgb = imcomplement (cmy);
```

[例9.1]从RGB转换到CMY。

```
I = imread('../plane.bmp');
cmy = imcomplement(I);
figure, imshow(cmy);
```

上面的代码将RGB图像plane.bmp转换到CMY空间下，结果如图9.7所示。为了观察效果， 9.7（b）中的图像是把 CMY分量直接以RGB格式来显示的，而事实上转换只是同一幅图像在不同色彩空间中的两种表示，就像同一幅图像可以在空间域和频域中描述一样。



（a）原图像 plane.bmp　　（b）转换后的 CMY 图像（以 RGB 的格式来显示）

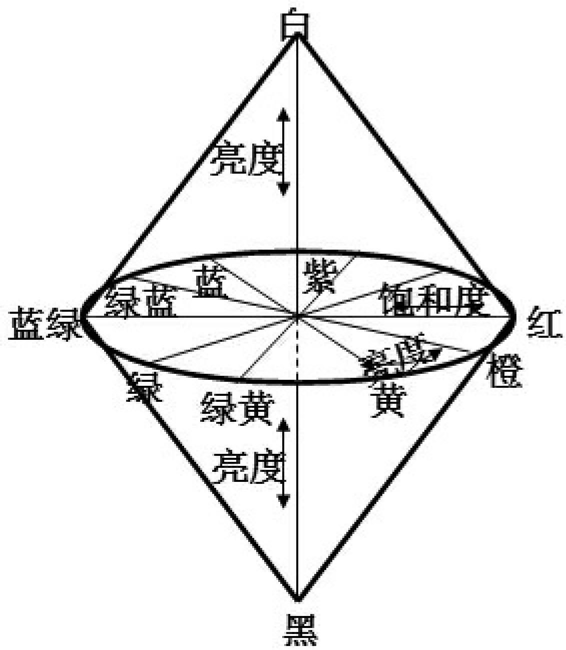
▲图9.7 RGB转CMY效果图

9.2.3 HSI模型

HSI 模型是从人的视觉系统出发，直接使用颜色三要素色调（Hue）、饱和度（Saturation）和亮度（Intensity，有时也翻译作密度或灰度）来描述颜色。

- 亮度是指人眼感觉光的明暗程度。光的能量越大，亮度越大。
- 色调是彩色最重要的属性，决定颜色的本质，由物体反射光线中占优势的波长来决定，不同的波长产生不同的颜色感觉，我们叫某一种颜色为红、橙、黄，这就是说我们在规定一种色调。
- 饱和度是指颜色的深浅和浓淡程度，饱和度越高，颜色越深。饱和度的深浅和白色的比例有关，白色比例越多，饱和度越低。

HSI彩色空间可以用一个圆锥空间模型来描述，如图9.8所示。我们通常把色调和饱和度统称为色度，用来表示颜色的类别与深浅程度。在图9.8中圆锥中间的横截面圆就是色度圆，而圆锥向上或向下延伸的便是亮度分量的表示。



▲图9.8 HSI模型示意图

由于人的视觉对亮度的敏感程度远强于对颜色浓淡的敏感程度，为了便于颜色处理和识别，人的视觉系统经常采用HSI彩色空间，它比RGB彩色空间更符合人的视觉特性。此外，由于HSI空间中亮度和色度具有可分离特性，使得图像处理和机器视觉中大量灰度处理算法都可在HSI彩色空间中方便地使用。

HSI彩色空间和RGB彩色空间只是同一物理量的不同表示法，因而它们之间存在着转换关系。下面将介绍一下，RGB到HSI的彩色转换和HSI到RGB的彩色转换。

1．从**RGB**到**HSI**的彩色转换及其实现

给定一幅RGB格式的图像，每一个RGB像素和H分量可用下面的公式得到。

$$H=\begin{cases}\theta,(B\leqslant G)\\360-\theta,(B>G)\end{cases}\qquad（9-1）$$

此处

$$\theta = \arccos\left\{\frac{\frac{1}{2}\left[(R-G)+(R-B)\right]}{\left[(R-G)^2+(R-G)(G-B)\right]^{1/2}}\right\}$$

饱和度分量由下式给出。

$$S = 1 - \frac{3}{(R+G+B)}\left[\min(R, G, B)\right] \tag{9-2}$$

最后，强度分量为

$$I = \frac{1}{3}(R+G+B) \tag{9-3}$$

假定RGB值归一化为[0，1]范围内，色调可以用式（9-1）得到的值除以360°归一化为[0，1]范围内，而其他两个HSI分量已经在[0，1]范围之内。

● Matlab实现

下面给出一个实现 RGB 转换到 HSI 的Matlab函数rgb2hsi，读者可以在金羽图书论坛（http://bbs.book95.com）的“金羽图书与答疑”板块与本书同名的主题帖子附件的第9章的code目录中找到它。

```
function hsi = rgb2hsi(rgb)
% hsi=rgb2hsi(rgb)把一幅RGB图像转换为HSI图像，
% 输入图像是一个彩色像素的M×N×3的数组，
% 其中每一个彩色像素都在特定空间位置的彩色图像中对应红、绿、蓝3个分量。
% 假如所有的RGB分量是均衡的，那么HSI转换就是未定义的。
```

```
% 输入图像可能是double（取值范围是[0, 1]），uint8或 uint16。
%
% 输出HSI图像是double，
% 其中hsi(:, :, 1)是色度分量，它的范围是除以2*pi后的[0, 1]；
% hsi(:, :, 2)是饱和度分量，范围是[0, 1]；
% hsi(:, :, 3)是亮度分量，范围是[0, 1]。
% 抽取图像分量
rgb = im2double(rgb);
r = rgb(:, :, 1);
g = rgb(:, :, 2);
b = rgb(:, :, 3);
% 执行转换方程
num = 0.5*((r - g) + (r - b));
den = sqrt((r - g).^2 + (r - b).*(g - b));
theta = acos(num./(den + eps)); %防止除数为0
H = theta;
H(b > g) = 2*pi - H(b > g);
H = H/(2*pi);
num = min(min(r, g), b);
den = r + g + b;
den(den == 0) = eps; %防止除数为0
S = 1 - 3.* num./den;
H(S == 0) = 0;
I = (r+g+b)/3;
% 将3个分量联合成为一个HSI图像
hsi = cat(3, H, S, I);
```

下面是一个调用rgb2hsi函数的程序段，它将RGB图像plane.bmp转

换至HSI空间。

```
>>figure;
subplot(1,2,1);
rgb=imread('plane.bmp');
imshow(rgb);title('rgb');
subplot(1,2,2);
hsi=rgb2hsi(rgb);
imshow(hsi);title('hsi');
```

转换效果如图9.9所示。

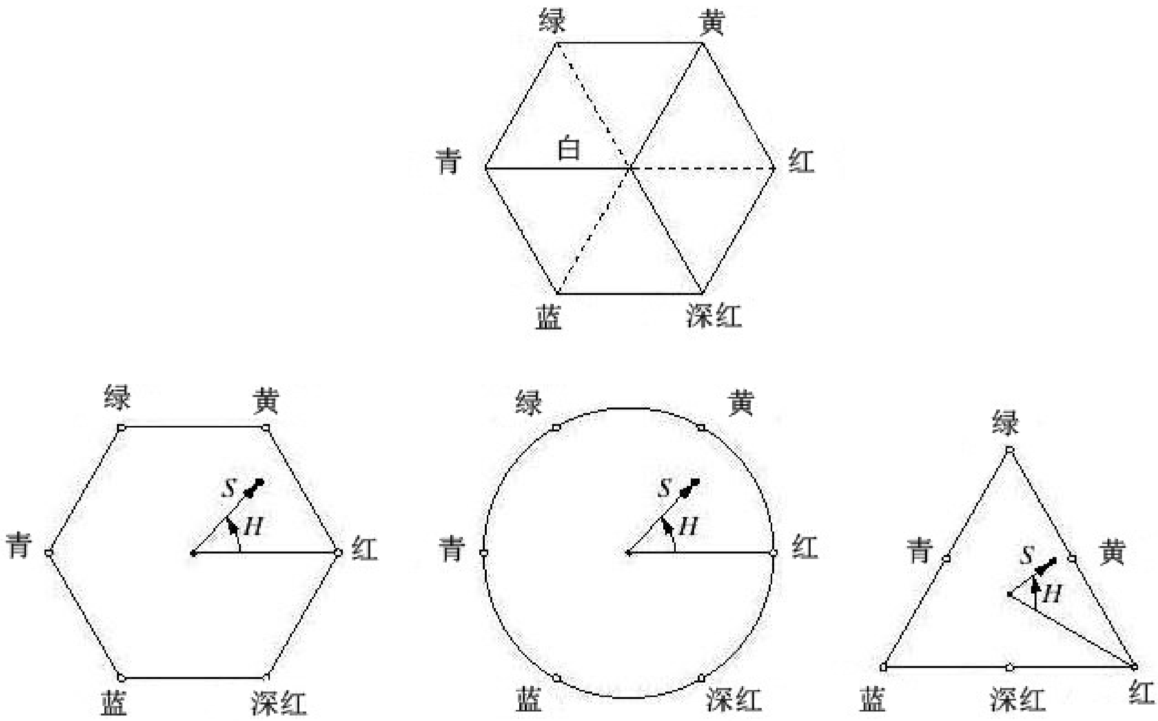


（a）RGB 原图　　（b）转换后 HSI 图（以 RGB 格式显示）

▲图9.9 RGB转HSI效果图

2．从**HSI**到**RGB**的彩色转换及其实现

在[0，1]内给出HSI值，现在要在相同的值域找到RGB值，可利用H值公式。在原始色分割中有3个相隔120°的扇形，如图9.10所示。从H乘以360°开始，这时色调值返回原来的[0，360]的范围。



▲图9.10 HSI模型中的色调和饱和度

RG扇形($0° \le H < 120°$)：当H位于这一扇形区时，RGB分量由下式给出。

$$B = I(1 - S)$$

$$R = I\left[1 + \frac{S\cos H}{\cos(60° - H)}\right]$$

$$G = 3I - (R + B)$$

GB扇区($120° \le H < 240°$)：如果给定的H值在这一扇区，首先从H中减去120°，即

$$H = H - 120°$$

然后RGB分量为

$$R = I(1-S)$$

$$G = I\left[1+\frac{S\cos H}{\cos(60°-H)}\right]$$

$$B = 3I-(R+G)$$

BR扇区(240°≤H<360°)：最后，如果H在这一扇区，从H中减去240°。

$$H = H-240°$$

然后RGB分量为

$$G = I(1-S)$$

$$B = I\left[1+\frac{S\cos H}{\cos(60°-H)}\right]$$

$$R = 3I-(G+B)$$

● Matlab实现

下面给出一个实现 HSI 转换到RGB的Matlab函数 hsi2rgb， 读者可以在金羽图书论坛（http://bbs.book95.com）的“金羽图书与答疑”板块与本书同名的主题帖子附件的第9章的code目录中找到它。

```
function rgb = hsi2rgb(hsi)
% rgb = hsi2rgb(hsi)把一幅HSI图像转换为RGB图像，
% 其中hsi(:, :, 1)是色度分量，它的范围是除以2*pi后的[0, 1]；
% hsi(:, :, 2)是饱和度分量，范围是[0, 1]；
% hsi(:, :, 3)是亮度分量，范围是[0, 1]。
%
% 输出图像分量：
```

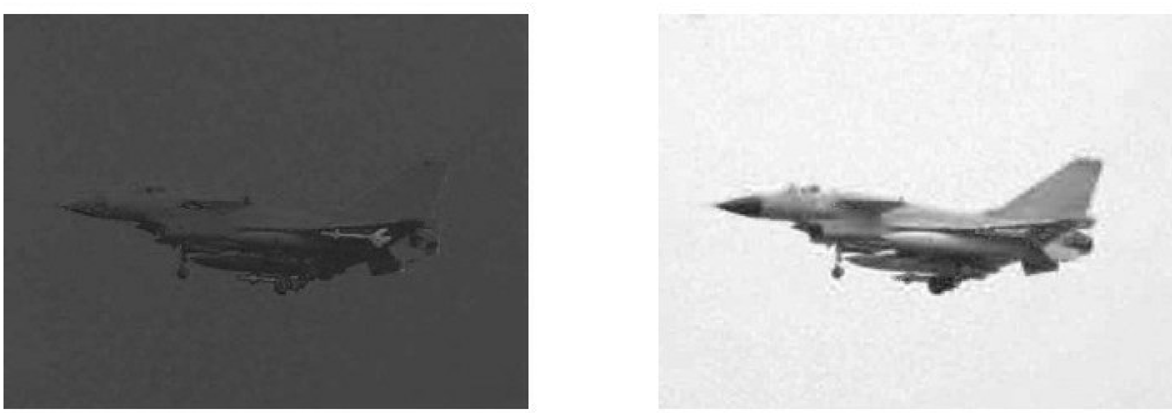
```
% rgb(:, :, 1)为红；
% rgb(:, :, 2)为绿；
% rgb(:, :, 3)为蓝。
% 抽取图像分量
hsi = im2double(hsi);
H = hsi(:, :, 1) * 2 * pi;
S = hsi(:, :, 2);
I = hsi(:, :, 3);
% 执行转换方程
R = zeros(size(hsi, 1), size(hsi, 2));
G = zeros(size(hsi, 1), size(hsi, 2));
B = zeros(size(hsi, 1), size(hsi, 2));
% RG扇形(0 <= H < 2*pi/3)
idx = find( (0 <= H) & (H < 2*pi/3));
B(idx) = I(idx) .* (1 - S(idx));
R(idx) = I(idx) .* (1 + S(idx) .* cos(H(idx)) ./ ...
cos(pi/3 - H(idx)));
G(idx) = 3*I(idx) - (R(idx) + B(idx));
% BG扇形(2*pi/3 <= H < 4*pi/3)
idx = find( (2*pi/3 <= H) & (H < 4*pi/3) );
R(idx) = I(idx) .* (1 - S(idx));
G(idx) = I(idx) .* (1 + S(idx) .* cos(H(idx) - 2*pi/3) ./ ...
cos(pi - H(idx)));
B(idx) = 3*I(idx) - (R(idx) + G(idx));
% BR扇形
idx = find( (4*pi/3 <= H) & (H <= 2*pi));
G(idx) = I(idx) .* (1 - S(idx));
```

```
B(idx) = I(idx) .* (1 + S(idx) .* cos(H(idx) - 4*pi/3) ./ ...
cos(5*pi/3 - H(idx)));
R(idx) = 3*I(idx) - (G(idx) + B(idx));
% 将3个分量联合成为一个RGB图像
rgb = cat(3, R, G, B);
rgb = max(min(rgb, 1), 0);
```

下面是一个调用hsi2rgb()函数的程序段，实现了HSI到RGB的彩色转换。

```
>>figure;
subplot(1,2,1);
hsi=imread('hsi.jpg'); %RGB图像plane.bmp转换至his空间后的图像
imshow(hsi);title('hsi');
subplot(1,2,2);
rgb=hsi2rgb(hsi);
imshow(rgb);title('rgb');
```

转换前后的效果如图9.11所示。



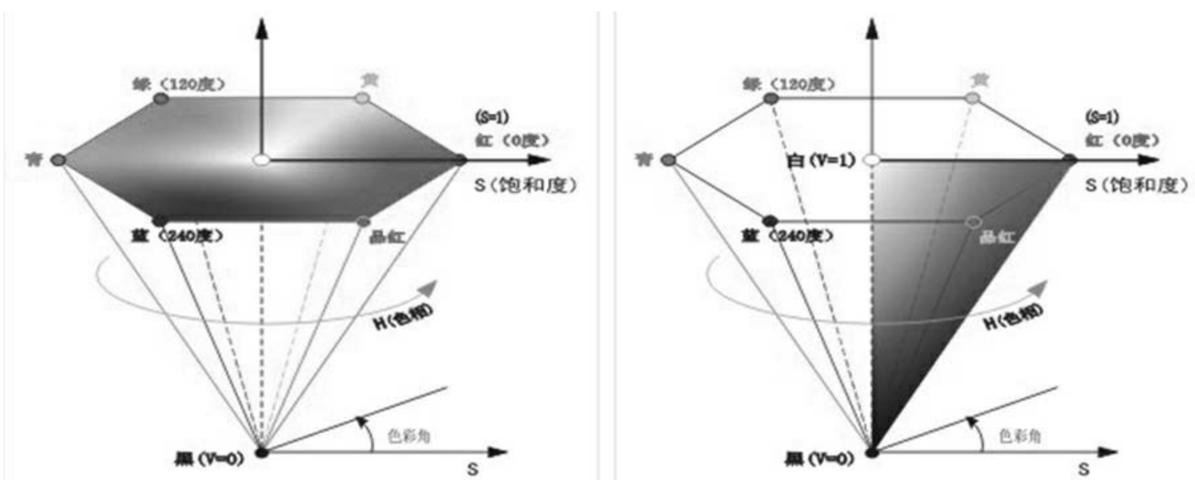
（a）HSI 原图（以 RGB 格式显示）　　（b）转换后 RGB 图

▲图9.11 HSI转 RGB效果图

9.2.4 HSV模型

HSV 模型是人们用来从调色板或颜色轮中挑选颜色（如颜料、墨水等）所采用的彩色系统之一。HSV（hue, saturation, value）表示色调（也称色相）、饱和度和数值。该系统比RGB更接近于人们的经验和对彩色的感知。在绘画术语中，色调、饱和度和数值用色泽、明暗和调色来表达。

HSV模型空间可以用一个倒立的六棱锥来描述，如图9.12所示。顶面是一个正六边形，沿H方向表示色相的变化，从0°～360°是可见光的全部色谱。六边形的6个角分别代表红、黄、绿、青、蓝、品红6个颜色的位置，每个颜色之间相隔60°。由中心向六边形边界（S方向）表示颜色的饱和度S变化，S的值由0～1变化，越接近六边形外框的颜色饱和度越高，处于六边形外框的颜色是饱和度最高的颜色，即S=1；处于六边形中心的颜色饱和度为0，即S=0。六棱锥的高（也即中心轴）用V表示，它从下至上表示一条由黑到白的灰度，V的底端是黑色，V=0；V的顶端是白色， V=1。



▲图9.12 HSV的六棱锥模型

1．从**RGB**到**HSV**的转换及其实现

假设所有的颜色值都已经归一化到范围[0,1]。在RGB3个分量中，

设定最大的为MAX，最小的为MIN。RGB到HSV的转换公式为

$$H=\begin{cases}\dfrac{G-B}{MAX-MIN}\times 60^\circ,(R=MAX)\\ \left(2+\dfrac{B-R}{MAX-MIN}\right)\times 60^\circ,(G=MAX)\\ \left(4+\dfrac{R-G}{MAX-MIN}\right)\times 60^\circ,(B=MAX)\end{cases}$$

$$S=\frac{MAX-MIN}{MAX}$$

$$V=MAX$$

计算结果中，H的值范围为0°～360°，S和V值范围为0～1。

如果MAX=MIN，H=没有色彩，是纯灰色。

如果H<0°，则H值得再加上360°。

如果MAX=0，S=0就是没有色彩。

如果V=0则是纯黑色。

● Matlab实现

在Matlab中RGB转为HSV的函数为rgb2hsv，其调用方式如下。

hsv=rgb2hsv(rgb);

输入的RGB图像可以是uint8、uint16或double类型的。

输出图像hsv为M×N×3的double类型。

转换后的效果如图9.13所示。



（a）RGB 原图　（b）转换后 HSV 图（以 RGB 的格式显示）

▲图9.13 RGB转HSV效果图

2．从**HSV**到**RGB**的转换及其实现

假设HSV颜色值已转换为这个范围：H的值为0°～360°，S和V值为0～1。则HSV到RGB的转换公式为

假如S=0，则R、G、B值的计算公式为

$$R = G = B = V$$

假如S≠0，则R、G、B值的计算公式为

$$i = \left\lfloor \frac{H}{60} \right\rfloor$$

$$f = \frac{H}{60} - i$$

$$p = V(1 - S)$$

$$q = V(1 - fS)$$

$$t = V(1 - (1 - f)S)$$

$$R = V, G = t, B = p, (i = 0)$$

$$R = q, G = V, B = p, (i = 1)$$

$$R = q, G = V, B = t, (i = 2)$$

$$R = p, G = q, B = V, (i = 3)$$

$$R = t, G = p, B = V, (i = 4)$$

$$R = V, G = p, B = q, (i = 5)$$

计算结果中，R、G和B值范围为0～1。

● Matlab实现

在Matlab中HSV转RGB的函数为hsv2rgb，其调用方式为

rgb = hsv2rgb(hsv); //输入输出图像均为double类型

转换后的效果如图9.14所示。



（a）HSV 图



（b）转换后 RGB 图

▲图9.14 HSV转 RGB效果图

与HSI转RGB效果相比，HSV转RGB的效果明显更好，并且可以快速高效地实现。所以，在实际中RGB和HSV的可逆转换往往比RGB和HSI的可逆转换使用得更多。

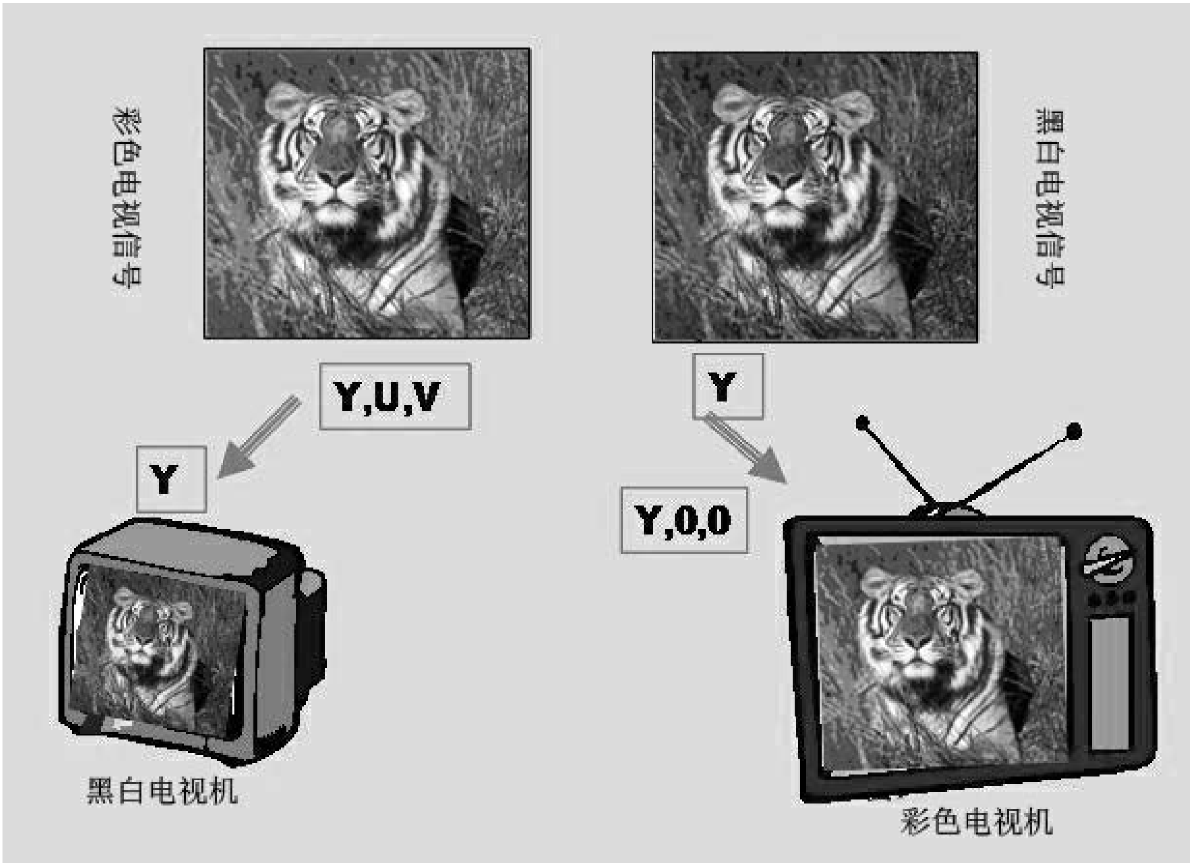
9.2.5 YUV模型

●UV是被欧洲电视系统所采用的一种颜色编码方法，是PAL和SECAM模拟彩色电视制式采用的颜色空间。其中的Y、U、V几个字母不是英文单词的组合词，Y代表亮度，U、V代表色差，是构成彩色的两个分量。

彩色图像信号经分色和分别放大校正得到RGB图像，再经过矩阵变换电路得到亮度信号Y和两个色差信号R－Y、B－Y，最后发送端将亮度和色差3个信号分别进行编码，用同一信道发送出去。这就是我们常用的YUV模型。

采用YUV模型的一个主要优势是它的亮度信号Y和色度信号U、V是分离的。如果只有Y信号分量而没有U、V分量，那么这样表示的图就是黑白灰度图。彩色电视采用YUV空间正是为了用亮度信号 Y 解决彩色电视机与黑白电视机的兼容问题，使黑白电视机也能接收彩色信号，

如图9.15所示。



▲图9.15 YUV模型应用示意图

当白光的亮度用 Y 来表示时，它和红、绿、蓝三色光的关系可用如下式的方程描述。Y=0.299R+0.587G+0.114B，这就是常用的亮度公式。色差U、V是由B－Y、R－Y按不同比例压缩而成的。如果要由YUV空间转化成RGB空间，只要进行相反的逆运算即可。

1．从**RGB**到**YUV**的转换及其实现

$$Y = 0.299R + 0.587G + 0.114B$$

$$U = 0.567(B - Y)$$

$$V = 0.713(R - Y)$$

● Matlab实现

我们编写了 rgb2yuv 函数，实现从 rgb 向 yuv 的转换。读者可以在金羽图书论坛（http://bbs.book95.com）的“金羽图书与答疑”板块与本书同名的主题帖子附件的第9章的code目录中找到文件rgb2yuv.m。

```
function yuv = rgb2yuv(rgb)
% yuv = rgb2yuv(rgb)把一幅RGB图像转换为YUV图像，
% 输入图像是一个彩色像素的M×N×3的数组，
% 其中每一个彩色像素都在特定空间位置的彩色图像中对应红、绿、蓝3个分量。
% 假如所有的RGB分量是均衡的，那么HSI转换就是未定义的。
% 输入图像可能是double（取值范围是[0, 1]），uint8或 uint16。
%
% 输出YUV图像是uint8。
rgb = im2double(rgb);
r = rgb(:, :, 1);
g = rgb(:, :, 2);
b = rgb(:, :, 3);
% 执行转换函数
y = 0.299*r + 0.587*g + 0.114*b;
u = 0.567*(b - y);
v = 0.713*(r - y);
% 防止溢出
if(y < 0)
  y = 0;
end;
if(y > 1.0)
  y = 1.0;
```

```
end;
if(u < 0)
   u = 0;
end;
if(u > 1.0)
   u = 1.0;
end;
if(v < 0)
   v = 0;
end;
if(v > 1.0)
   v = 1.0;
end;
% 联合yuv，并转成uint8类型
y = y*255;
u = u*255;
v = v*255;
yuv = cat(3, y, u, v);
yuv = uint8(yuv);
```

调用rgb2yuv()函数转换后的效果如图9.16所示。



（a）RGB 原图



（b）转换后 YUV 图（以 RGB 的格式显示）

▲图9.16 RGB转 YUV效果图

2．从**YUV**到**RGB**的转换及其实现

$$R = Y + 1.402V$$
$$G = Y - 0.344U - 0.714V$$
$$B = Y + 1.772U$$

● Matlab实现

我们编写了 yuv2rgb 函数，实现从yuv向rgb的转换。读者可以在金羽图书论坛（http://bbs.book95.com）的“金羽图书与答疑”板块与本书同名的主题帖子附件的第9章的code目录中找到文件rgb2yuv.m。

```
function rgb = yuv2rgb(yuv)
% yuv = rgb2yuv(rgb)把一幅RGB图像转换为YUV图像，
% 输入图像是一个彩色像素的M×N×3的数组，
% 其中每一个彩色像素都在特定空间位置的彩色图像中对应红、绿、蓝3个分量。
% 假如所有的RGB分量是均衡的，那么HSI转换就是未定义的。
% 输入图像可能是double（取值范围是[0, 1]），uint8或 uint16。
%
```

```
% 输出YUV图像是uint8。
yuv = im2double(yuv);
y = yuv(:, :, 1);
u = yuv(:, :, 2);
v = yuv(:, :, 3);
% 执行转换函数
r = y + 1.402*v;
g = y - 0.344*u - 0.714*v;
b = y + 1.772*u;
% 防止溢出
if(r < 0)
   r = 0;
end;
if(r > 1.0)
   r = 1.0;
end;
if(g < 0)
   g = 0;
end;
if(g > 1.0)
   g = 1.0;
end;
if(b < 0)
   b = 0;
end;
if(b > 1.0)
   b = 1.0;
```

```
end;
% 联合rgb
r = r*255;
g = g*255;
b = b*255;
rgb = cat(3, r, g, b);
rgb = uint8(rgb);
```

转换后的效果如图9.17所示。



（a）YUV 图（以 RGB 格式显示）　　（b）转换后 RGB 图

▲图9.17 YUV转 RGB效果图

9.2.6 YIQ模型

YIQ模型是北美NTSC彩色制式，主要用于美国的电视系统。这种形式和欧洲的YUV模型有相同的优势：灰度信息和彩色信息是分离的。YIQ模型中，Y代表亮度，I代表色调，Q则代表饱和度。其中，亮度表示灰度，而色调和饱和度则存储彩色信息。

1．从**RGB**到**YIQ**的转换及其实现

转换公式为

$$\begin{bmatrix} Y \\ I \\ Q \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ 0.596 & -0.274 & -0.322 \\ 0.211 & -0.523 & 0.312 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

上式中，第一行和为1，下两行和分别为0。

● Matlab实现

Matlab函数rgb2ntsc可实现从rgb到ntsc的转换，其调用形式为

●iq = rgb2ntsc(rgb);

输入的RGB图像可以是uint8、uint16或double类型的。

输出图像为M×N×3的double类型。

转换后的效果如图9.18所示。



（a）RGB 原图



（b）转换后 YIQ 图（以 RGB 格式显示）

▲图9.18 RGB转 YIQ效果图

2．从**YIQ**到**RGB**的转换及其实现

转换公式为

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1.000 & 0.956 & 0.621 \\ 1.000 & -0.272 & -0.647 \\ 1.000 & -1.106 & 1.703 \end{bmatrix} \begin{bmatrix} Y \\ I \\ Q \end{bmatrix}$$

● Matlab实现

Matlab函数ntsc2rgb可实现从ntsc到rgb的转换，其调用形式为

rgb = ntsc2rgb(yiq);

其中，输入和输出图像均为double类型。

转换后的效果如图9.19所示。



（a）YIQ 图　　（b）转换后 RGB 图

▲ 图9.19 YIQ转 RGB效果图

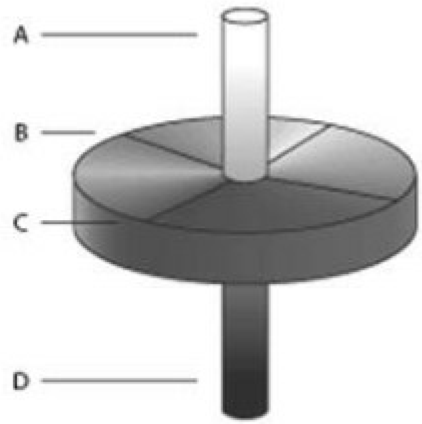
9.2.7 Lab模型简介

Lab模型是由CIE（国际照明委员会）制定的一种彩色模式。这种模型与设备无关，它弥补了RGB模型和CMYK模型必须依赖于设备颜色特性的不足。此外，自然界中任何色彩都可在Lab空间表达出来，这就意味着RGB以及CMYK所能描述的颜色信息在Lab中都能得以影射。

Lab颜色空间如图9.20所示。其中L表示亮度；a的正数代表红色，负端代表绿色；b的正数代表黄色，负端代表蓝色。

由于Lab模型与设备无关的特点，使其在彩色图像检索中应用较为广泛。另外，当希望在图像的处理中保留尽量宽阔的色域和丰富的色彩时，可以选择在Lab模型下进行工作，处理后再根据输出的需要转换成RGB模型（显示用）或CMYK模型（打印及印刷用）。这样做的最大好

处是它能够在最终的设计成果中，获得比任何色彩模型都更加优质的色彩。



A — 亮度为100（白）；B — 深绿到亮粉；
C — 亮蓝到焦黄；D — 亮度为0（黑）

▲图9.20 Lab模型示意图

9.3 全彩色图像处理基础

本节要介绍全彩色图像处理技术，面对各种各样的图像处理任务怎样处理全彩色图像。通常，全彩色图像处理技术总的可以分为两大类。

（1）对3个平面分量单独处理，然后将分别处理过的3个分量合成彩色图像。对每个分量的处理技术可以应用到对灰度图像处理的技术上。但是这种通道式的独立处理技术忽略了通道间的相互影响。

（2）直接对彩色像素进行处理。因为全彩色图像至少有3个分量，彩色像素实际上是一个向量。直接处理就是同时对所有分量进行无差别的处理。这时彩色图像的3个分量用向量形式表示，即对彩色图像上任一点的像素c(x, y)，有

$$c(x,\ y)\ =\ [R(x,\ y);G(x,\ y);B(x,\ y)]$$

那么对像素点(x, y)处理的操作实际上是同时对R、G、B这3个分量操作。不过通常大多数图像处理技术都是指对每个分量的单独处理。接下来将讲述全彩色图像处理的两个常用技术：彩色补偿和彩色平衡。

9.3.1 彩色补偿及其Matlab实现

有些图像处理任务的目标是根据颜色分离出不同的类型的物体。但由于常用的彩色成像设备具有较宽且相互覆盖的光谱敏感区，加之待拍摄图像的染色是变化的，所以很难在3个分量图中将物体分离出来，这种现象称为颜色扩散。彩色补偿的作用就是通过不同的颜色通道提取不同的目标物。

1．理论基础

彩色补偿算法描述如下。

（1）在画面上找到主观视觉看是纯红、纯绿、纯蓝的3个点（若可根据硬件知道频段的覆盖则无须这样做。）

$$P_1 = (R_1,G_1,B_1) \qquad P_1^* = (R^*,0,0)$$
$$P_2 = (R_2,G_2,B_2) \quad \xrightarrow{\text{它们的理想值为}} \quad P_2^* = (0,\ G^*,0)$$
$$P_3 = (R_3,G_3,B_3) \qquad P_3^* = (0,0,\ B^*)$$

（2）计算r^*、g^*、B^*的值。考虑到彩色补偿之后图像的亮度不变，则对r^*、g^*、B^*的计算为

$$R^* = 0.30*\ \mathrm{R}_1 + 0.59*\ \mathrm{G}_1 + 0.11*\ \mathrm{B}_1$$
$$G^* = 0.30*\ \mathrm{R}_2 + 0.59*\ \mathrm{G}_2 + 0.11*\ \mathrm{B}_2$$
$$B^* = 0.30*\ \mathrm{R}_3 + 0.59*\ \mathrm{G}_3 + 0.11*\ \mathrm{B}_3$$

（3）构造变换矩阵。将所取到的3个点的RGB值分别按如下所示构

造出的彩色补偿前和补偿后的2个矩阵A_1和A_2。

$$A_1=\begin{bmatrix} R_1 & R_2 & R_3 \\ G_1 & G_2 & G_3 \\ B_1 & B_2 & B_3 \end{bmatrix}$$

$$A_2=\begin{bmatrix} R^* & 0 & 0 \\ 0 & G^* & 0 \\ 0 & 0 & B^* \end{bmatrix}$$

（4）进行彩色补偿。设 $S(x,y)=\begin{bmatrix} R_s(x,y) \\ G_s(x,y) \\ B_s(x,y) \end{bmatrix}$，$F(x,y)=\begin{bmatrix} R_F(x,y) \\ G_F(x,y) \\ B_F(x,y) \end{bmatrix}$分别为新、旧图像的像素值。

则$S(x,y)=C^{-1}*F(x,y)$。其中，$C=A_1*A_2^{-1}$。

2．Matlab实现

根据上述算法编写的彩色补偿 Matlab 实现如下。读者可以在金羽图书论坛（http://bbs.book95.com）的“金羽图书与答疑”板块与本书同名的主题帖子附件的第9章的code目录中找到文件compensate.m。

```
% compensate.m
% 彩色补偿
im=double(imread('plane.bmp'));
subplot(1,2,1);
imshow(uint8(im));
title('原始图');
[m,n,p]=size(im);
```

```
[h1,k1]=min(255-im(:,:,1)+im(:,:,2)+im(:,:,3));
[j1,minx]=min(h1);
 i1=k1(j1);%提取图像中最接近红色的点，其在im中的坐标为i1,j1
 r1=im(i1,j1,1);
 g1=im(i1,j1,2);
 b1=im(i1,j1,3);
R=0.30*r1+0.59*g1+0.11*b1;
[h2,k2]=min(255-im(:,:,2)+im(:,:,1)+im(:,:,3));
[j2,minx]=min(h2);
 i2=k2(j2);%提取图像中最接近绿色的点，其在im中的坐标为i2,j2
 r2=im(i2,j2,1);
 g2=im(i2,j2,2);
 b2=im(i2,j2,3);
G=0.30*r2+0.59*g2+0.11*b2;
[h3,k3]=min(255-im(:,:,3)+im(:,:,1)+im(:,:,2));
[j3,minx]=min(h3);
 i3=k3(j3);%提取图像中最接近蓝色的点，其在im中的坐标为i3,j3
 r3=im(i3,j3,1);
 g3=im(i3,j3,2);
 b3=im(i3,j3,3);
B=0.30*r3+0.59*g3+0.11*b3;
A1=[r1 r2 r3
    g1 g2 g3
    b1 b2 b3];
A2=[R 0 0
    0 G 0
    0 0 B];
```

```
C=A1*inv(A2);
for i=1:m
  for j=1:n
    imR=im(i,j,1);
    imG=im(i,j,2);
    imB=im(i,j,3);
    temp=inv(C)*[imR;imG;imB];
    S(i,j,1)=temp(1);
    S(i,j,2)=temp(2);
    S(i,j,3)=temp(3);
  end
end
S=uint8(S);
subplot(1,2,2);
imshow(S);
title('补偿后');
```

采用上面的程序对RGB图像plane.bmp进行彩色补偿处理，结果如图9.21所示。



（a）原始图



（b）补偿后

▲图9.21 彩色补偿效果图

9.3.2 彩色平衡及其Matlab实现

一幅彩色图像数字化后，在显示时颜色经常看起来有些不正常。这是色通道的不同敏感度、增光因子和偏移量等原因导致的，称其为三基色不平衡。将之校正的过程就是彩色平衡。

1．理论基础

彩色平衡校正算法如下。

（1）从画面中选出两点颜色为灰色，设为

$$
\begin{aligned}
F_1 &= (R_1,G_1,B_1) \\
F_2 &= (R_2,G_2,B_2)
\end{aligned}
$$

（2）设以G分量为基准，匹配R和B分量，则

$$
\begin{aligned}
F_1 &= (R_1,G_1,B_1) \\
F_2 &= (R_2,G_2,B_2)
\end{aligned}
\quad \Rightarrow \quad
\begin{aligned}
F_1^* &= (R_1,G_1,B_1) \\
F_2^* &= (R_2,G_2,B_2)
\end{aligned}
$$

（3）由R_1^* =k1*R_1+k2和 R_2^* =k1*R_2+k2求出k1和k2；B_1^* =l1*b_1+l2和B_2^* =l1*b_2+l2求出l1和l2。

（4）用

$$R(x,y)^* = k1 * R(x,y) + k2$$

$$B(x,y)^* = l1 * B(x,y) + l2$$

$$G(x,y)^* = G(x,y)$$

处理后得到的图像就是彩色平衡后的图像。

2．**Matlab**实现

根据上述算法编写的彩色平衡 Matlab 实现如下。读者可以在金羽图书论坛（http://bbs.book95.com）的“金羽图书与答疑”板块与本书同名的主题帖子附件的第9章的code目录中找到文件colorBalance.m。

```
% colorBalance.m
% 彩色平衡
im=double(imread('plane.bmp'));
[m,n,p]=size(im);
F1=im(1,1,:);
F2=im(1,2,:);
F1_(1,1,1)=F1(:,:,2);
F1_(1,1,2)=F1(:,:,2);
F1_(1,1,3)=F1(:,:,2);
F2_(1,1,1)=F2(:,:,2);
F2_(1,1,2)=F2(:,:,2);
F2_(1,1,3)=F2(:,:,2);
K1=(F1_(1,1,1)-F2_(1,1,1))/(F1(1,1,1)-F2(1,1,1));
K2=F1_(1,1,1)-K1*F1(1,1,1);
L1=(F1_(1,1,3)-F2_(1,1,3))/(F1(1,1,3)-F2(1,1,3));
L2=F1_(1,1,3)-L1*F1(1,1,3);
for i=1:m
  for j=1:n
    new(i,j,1)=K1*im(i,j,1)+K2;
    new(i,j,2)=im(i,j,2);
    new(i,j,3)=L1*im(i,j,3)+L2;
  end
end
im=uint8(im);
```

```
new=uint8(new);
subplot(1,2,1);
imshow(im);
title('原始图');
subplot(1,2,2);
imshow(new);
title('平衡后');
```

采用上面的程序对RGB图像plane.bmp进行彩色平衡，结果如图9.22所示。



（a）原始图　　（b）平衡后

▲图9.22 彩色平衡效果图

第10章 形态学图像处理

形态学，即数学形态学（Mathematical Morphology），是图像处理中应用最为广泛的技术之一。其主要应用是从图像中提取对于表达和描绘区域形状有意义的图像分量，使后续的识别工作能够抓住目标对象最为本质（最具区分能力——most discriminative）的形状特征，如边界和连通区域等；同时像细化、像素化和修剪毛刺等技术也常常应用于图像的预处理和后处理中，成为图像增强技术的有力补充。

本章的知识和技术热点

- 二值图像的基本形态学运算，包括腐蚀、膨胀、开和闭。
- 二值形态学的经典应用，包括击中击不中变换、边界提取和跟踪、区域填充、提取连通分量、细化和像素化，以及凸壳。
- 灰度图像的形态学运算，包括灰度腐蚀、灰度膨胀、灰度开和灰度闭。

本章的典型案例分析

- 在人脸局部图像中定位嘴的中心。
- 显微镜下图像的细菌计数。
- 利用顶帽变换（top-hat）技术解决光照不均问题。

10.1 预备知识

在数字图像处理中，形态学是借助集合论的语言来描述的，本章后面的各节内容均以本节要介绍的集合论为基础。

在数字图像处理的形态学运算中，常常把一幅图像或者图像中一个我们感兴趣的区域称作集合，用大写字母A、B、C等表示；而元素通常是指一个单个的像素，用该像素在图像中的整型位置坐标z=(z1，z2)来

表示，这里$z\in Z^2$，其中Z^2为二元整数序偶对的集合。

下面介绍一些集合论中的重要关系。

1．集合与元素

属于：对于某一集合（图像区域）A，若点a在A之内，则称a为A的元素，a属于A，记作$a\in A$；反之，若点b不在A之内，称b不属于A，记作$b\notin A$，如图10.1（a）所示。

2．集合与集合

并集：$C=\{z\mid z\in A \text{ or } z\in B\}$，记作$C=A\cup B$，即A与B的并集C包含集合A与集合B的所有元素，如图10.1（b）所示。

交集：$C=\{z\mid z\in A \text{ and } x\in B\}$，记作$C=A\cap B$，即A与B的交集C包含同时属于集合A与集合B的所有元素，如图10.1（c）所示。

补集：$A^C=\{z\mid z\notin A\}$，即A的补集是不包含于A的所有元素组成的集合，如图 10.1（d）所示。

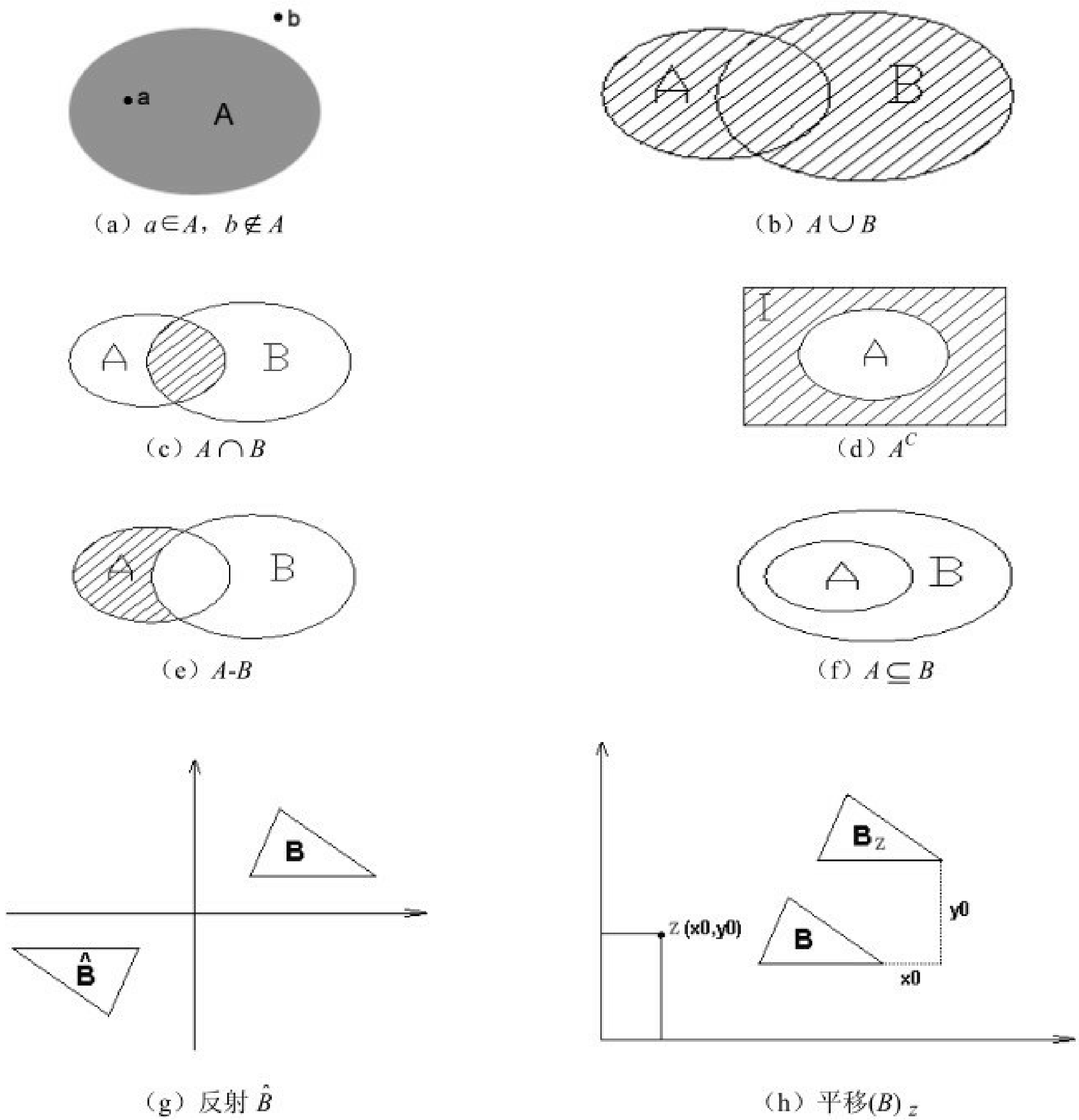
差集：$A-B=\{z\ z\in A, z\notin B\}=A\cap B^c$，即A与B的差集由所有属于A但不属于B的元素构成，如图10.1（e）所示。

包含：集合A的每1个元素都是另一个集合B的1个元素，则称A为B的子集，记作$A\subseteq B$，如图10.1（f）所示。

3．反射和平移

反射：又名对称，定义为$\hat{B}=\{z|z=-b, b\in B\}$，记作$\hat{B}$，如图10.1（g）所示。

平移：将集合B平移到点z=(z1，z2)，定义为$(B)_z=\{x|x=b+z, b\in B\}$，记作$(B)_z$，如图10.1 （h）所示。



▲图10.1 集合的关系和运算

4．结构元素（**structure element**）

设有两幅图像A、S。若A是被处理的对象，而S是用来处理A的，则称S为结构元素。结构元素通常都是一些比较小的图像，A与S的关系类似于滤波中图像和模板的关系。

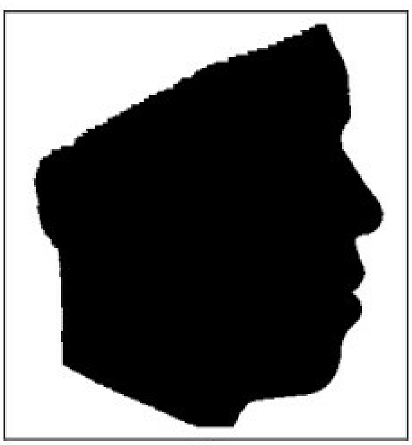
10.2 二值图像中的基本形态学运算

本节介绍几种二值图像的基本形态学运算，包括腐蚀、膨胀，以及开、闭运算。

由于所有形态学运算都是针对图像中的前景物体进行的，因而首先对图像前景和背景的认定给出必要的说明。

注意

对于大多数图像，一般相对于背景而言物体的颜色（灰度）更深，二值化之后物体会成为黑色；而背景则成为白色。因此我们通常更习惯于将物体用黑色（灰度值0）表示，而背景用白色（灰度值255）表示，本章所有的算法示意图都遵从这种约定；但Matlab在二值图像形态学处理中，默认情况下白色的(二值图像中灰度值为1的像素，或灰度图像中灰度值为255的像素)是前景（物体），黑色的为背景，因而本章涉及Matlab的所有程序实例又都遵从Matlab本身的这种前景认定习惯。所以图10.2 中所示的两幅图像对于这两种不同的认定习惯来说从形态学处理的意义上是等同的。



（a）



（b）

▲图10.2 两种常见的对于图像前景与背景的认定

实际上，无论以什么灰度值为前景和背景都只是一种处理上的习惯

而已，与形态学算法本身无关。例如对于上面的两幅图片，只需要在形态学处理之前先对图像反色，就可以在两种认定习惯之间自由切换。

10.2.1 腐蚀及其实现

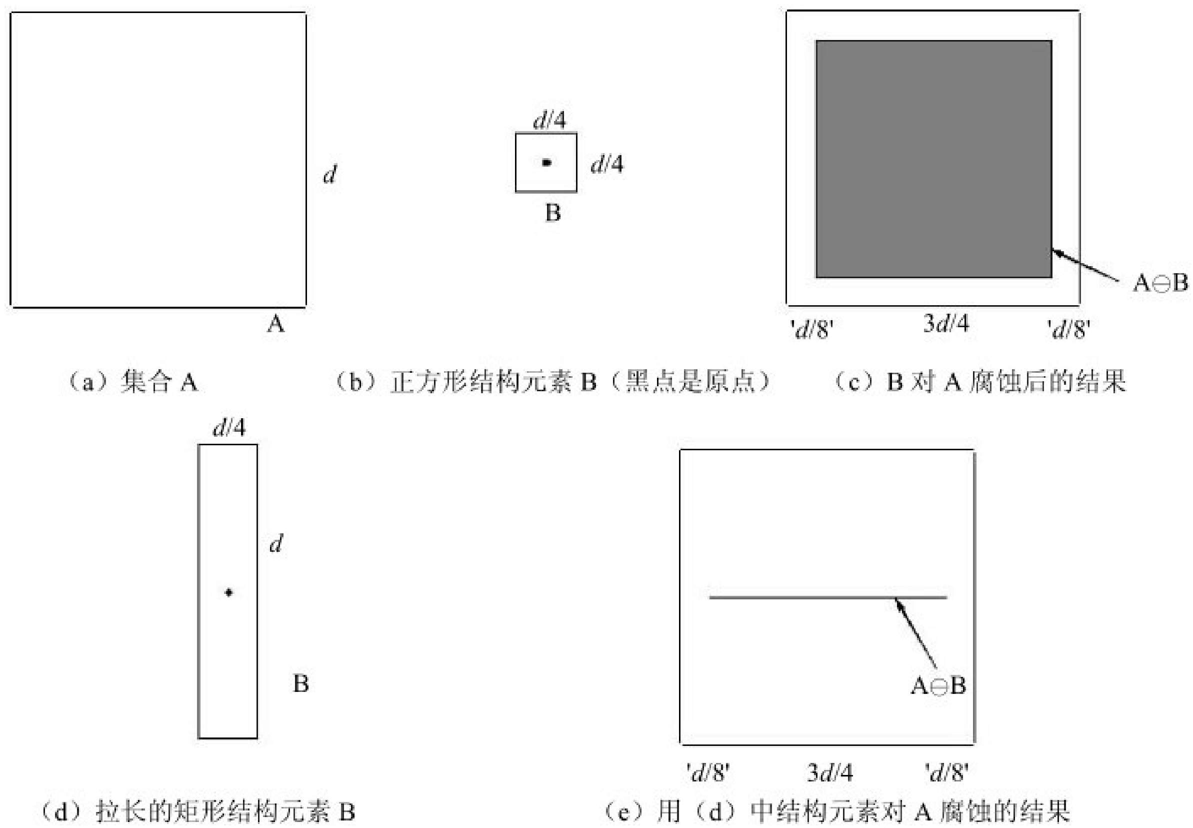
腐蚀和膨胀是两种最基本也是最重要的形态学运算，因为它们是后续要介绍的很多高级形态学处理的基础，很多其他的形态学算法都是由这两种基本运算复合而成的。下面先介绍腐蚀，在10.2.2中将介绍膨胀。

1．理论基础

对Z^2 上元素的集合A和S，使用S对A进行腐蚀，记作A $\ominus$ S，形式化地定义为

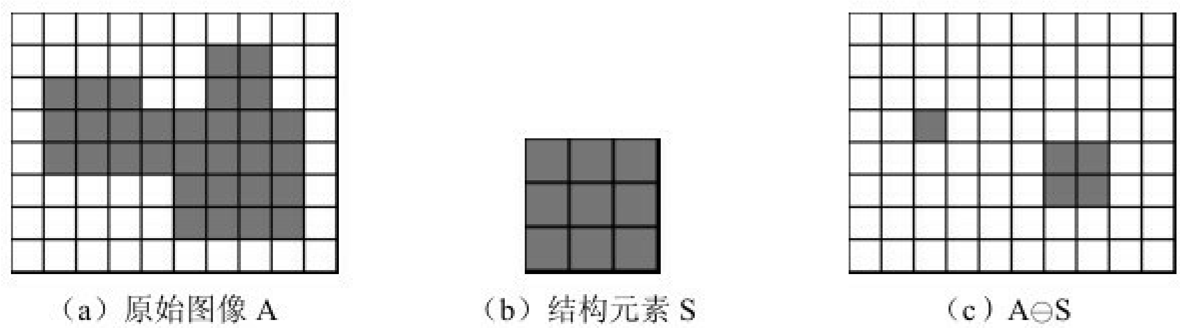
$$A \ominus S = \{z \mid (S)_z \subseteq A\} \quad (10\text{-}1)$$

让原本位于图像原点的结构元素S在整个Z^2平面上移动，如果当S的原点平移至z点时S能够完全包含于A中，则所有这样的z点构成的集合即为S对A的腐蚀图像，如图10.3所示。



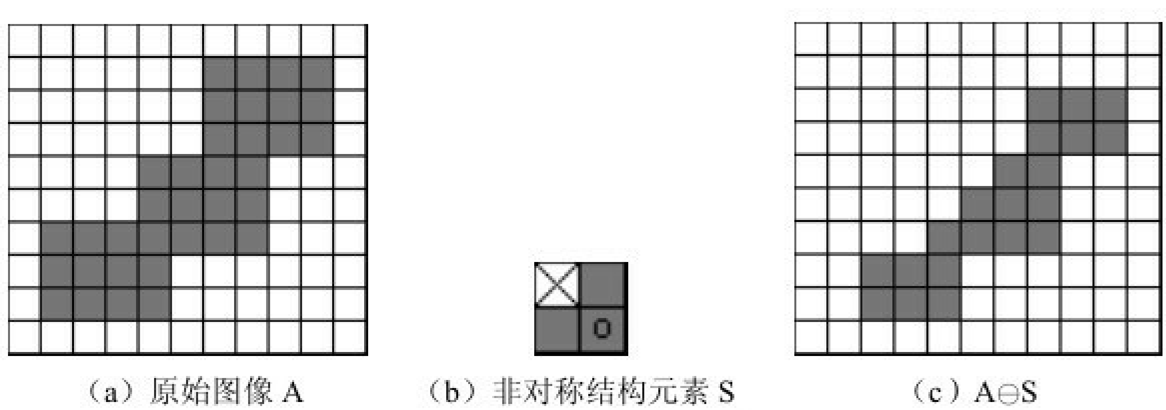
▲图10.3 腐蚀示意图（图中标出了所有图形的长和宽）

采用原点位于中心的3×3对称结构元素的腐蚀运算效果如图 10.4 所示。该效果是由我们在金羽图书论坛（http://bbs.book95.com）的“金羽图书与答疑”板块与本书同名的主题帖子附件中提供的MorphSimulator工具实现的，读者可以在第10章的Tools目录中找到此工具。本章所有的形态学运算效果均可由MorphSimulator模拟，该工具使用非常简单（详见同目录下的help.txt文件），它能够将形态学运算的效果放大化，用每1个方格代表1个像素，从而可以非常直观地感受运算效果，帮助初学者更快地理解各种形态学算法。



▲图10.4 腐蚀运算效果模拟

下面再来看一个非对称结构元素腐蚀的示例。如图10.5所示，结构元素的原点在图中以“O”标出。



▲图10.5 非对称结构元素的腐蚀

图（b）中像素格内的“O”表示原点，“X”表示该位置的值我们并不关心，稍后会对“X”的意义加以解释

形态学运算的运算结果不仅与结构元素的形状有关，还与结构元素的原点位置密切相关，请读者思考：同样对于图10.4中的图像A，结构元素形状不变，只是将原点放在第3行中间的那个元素上，会得到什么样的结果呢？

2．Matlab实现

Matlab中和腐蚀相关的两个常用函数为imerode和strel。

（1）imerode函数用于完成图像腐蚀，其常用调用形式如下。

I2 = imerode(I, SE);

参数说明

● I为原始图像，可以是二值或灰度图像（对应于灰度腐蚀）。

● SE是由strel函数返回的自定义或预设的结构元素对象。

返回值

● I2为腐蚀后的输出图像。

（2）strel函数可以为各种常见形态学运算生成结构元素SE，当生成供二值形态学使用的结构元素时，其调用形式为

SE = strel(shape,parameters);

参数说明

● shape指定了结构元素的形状，其常用合法取直如表10.1所示。

表10.1 常用平坦结构元素的shape参数

合法取值	功能描述
'arbitrary' 或为空	任意自定义的结构元素
'disk'	圆形结构元素
'square'	正方形结构元素
'rectangle'	矩形结构元素
'line'	线形的结构元素
'pair'	包含两个点的结构元素
'diamond'	菱形的结构元素
'octagon'	8 角形的结构元素

● parameters是和输入shape有关的参数。

返回值

● SE为得到的结构元素对象。

下面结合一些代表性的情况具体说明。

（1）SE = strel('arbitrary',NHOOD)返回一个由NHOOD指定的结构

元素。其中，NHOOD为一个只包含“0”和“1”的矩阵，规定了结构元素的形状。结构元素的中心位于 NHOOD 矩阵的中心位置，即 floor((size(NHOOD)+1)/2)。也可省略第 1个参数，写为SE = strel(NHOOD)的形式。

（2）SE = strel('disk',R)返回一个半径为R的圆形结构元素。

（3）SE = strel('pair',OFFSET)返回一个只包含2个“1”的结构元素。其中1个“1”位于原点，另1个“1”相对于原点的位置由OFFSET向量指定。OFFSET是一个长度为2的向量，OFFSET(1)是x方向的偏移量，OFFSET(2)是y方向的偏移量。

（4）SE = strel('rectangle',MN)返回一个高、宽均由向量MN指定的矩形结构元素。MN是一个长度为2的向量，MN(1)是结构元素的高度，MN(2)是结构元素的宽度。

提示

由于形态学运算中的结构元素通常都具有一定的尺寸，当结构元素位于图像边缘时，其中的某些元素很可能会位于图像之外，这时需要对于在边缘附近的操作单独处理，以避免引用到本不属于图像的无意义的值。这类似于滤波操作中的边界处理问题（5.2.2小节）。Matlab可以自动处理边界问题。

3．腐蚀的作用

顾名思义，腐蚀能够消融物体的边界，而具体的腐蚀结果与图像本身和结构元素的形状有关：如果物体整体上大于结构元素，腐蚀的结构是使物体变“瘦”一圈，这一圈到底有多大是由结构元素决定的；如果物体本身小于结构元素，则在腐蚀后的图像中物体将完全消失；如物体仅有部分区域小于结构元素（如细小的连通），则腐蚀后物体会在细连通处断裂，分离为两部分。下面例10.1说明了几种不同情况下的腐蚀效果。

[例10.1]不同结构元素的腐蚀效果。

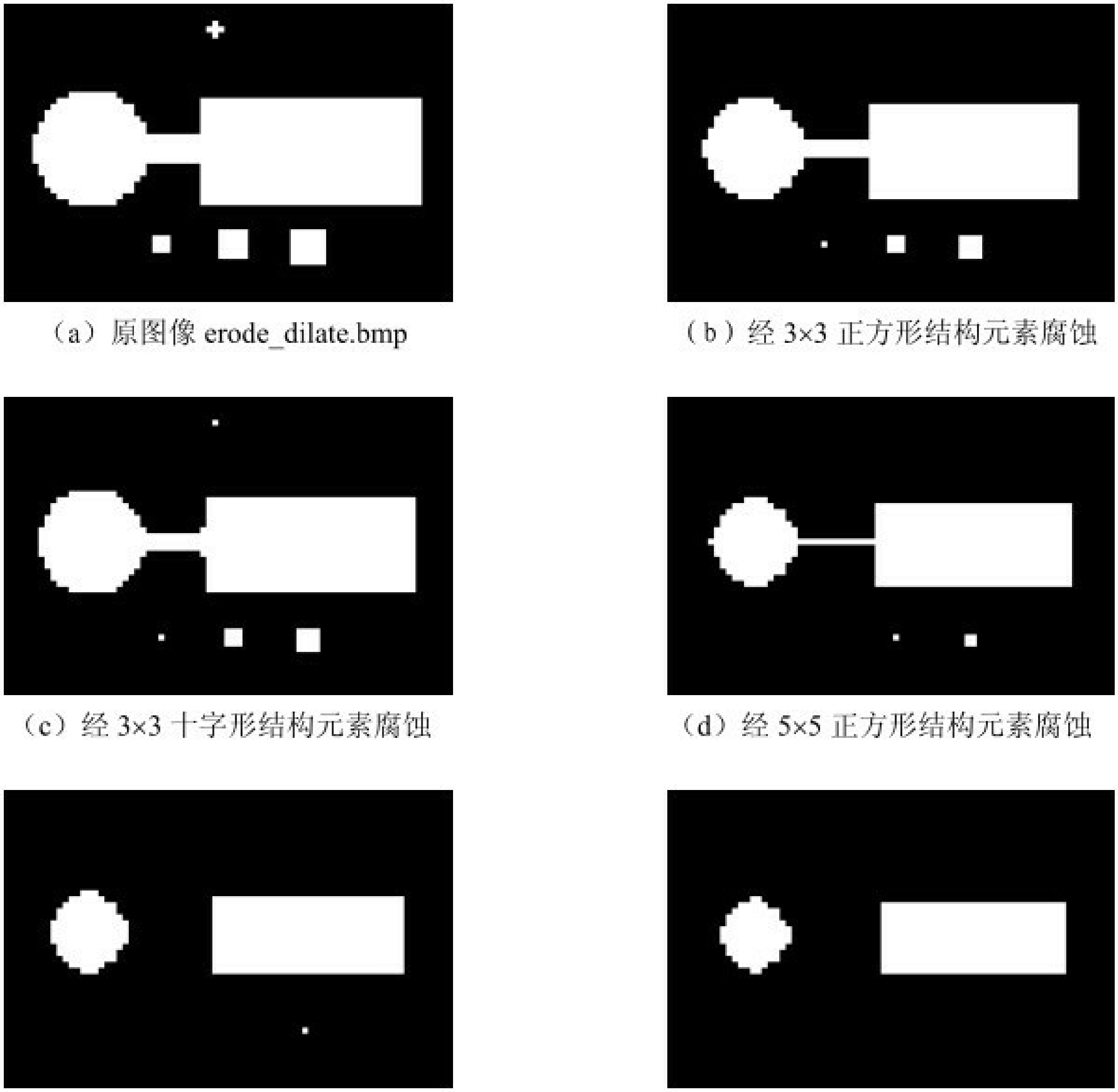
在图10.6（a）所示的原始图像中，两个主要的区域（圆形和矩形），中间用一个宽度为5的条形连通带，图像上中是一个横向、纵向长度均为3个像素的十字，位于图像底部的3个正方形的边长分别为3、5和6。

采用不同的结构元素对图10.6（a）进行腐蚀的Matlab程序如例10.1所示。

```
>>I = imread('../erode_dilate.bmp'); %读入8位灰度图像
%二值形态学处理中将灰度图像中所有非0值都看作是1，即前景物体
>> figure, imshow(I); %得到图10.6（a）
>> se = strel('square', 3) %3*3的正方形结构元素
se =
Flat STREL object containing 9 neighbors.
Neighborhood:
   1   1   1
   1   1   1
   1   1   1
>> Ib= imerode(I, se); %腐蚀
>> figure, imshow(Ib); %得到图10.6（b）
>>
>> se = strel([0 1 0; 1 1 1; 0 1 0]) %3*3的十字结构元素
 se =
Flat STREL object containing 5 neighbors.
Neighborhood:
   0   1   0
   1   1   1
   0   1   0
```

```
>> Ic= imerode(I, se); %腐蚀
>> figure, imshow(Ic); %得到图10.6（c）
>>
>> se = strel('square', 5) ;%5*5的正方形结构元素
>> Id= imerode(I, se); %腐蚀
>> figure, imshow(Id); %得到图10.6（d）
>>
>> se = strel('square', 6); %6*6的正方形结构元素
>> Ie= imerode(I, se); %腐蚀
>> figure, imshow(Ie); %得到图10.6（e）
>>
>> se = strel('square', 7) ;%7*7的正方形结构元素
>> If= imerode(I, se); %腐蚀
>> figure, imshow(If); %得到图10.6（f）
```

上述程序的运行效果如图10.6（b）～（f）所示。


（a）原图像 erode_dilate.bmp
（b）经 3×3 正方形结构元素腐蚀
（c）经 3×3 十字形结构元素腐蚀
（d）经 5×5 正方形结构元素腐蚀
（e）经 6×6 的正方形结构元素腐蚀
（f）经 7×7 正方形结构元素腐蚀

▲图10.6 腐蚀对不同物体的影响（为使图中前景物体更加明显，对原图放大了 3倍显示）

如图10.6（b）所示，原图像经过3×3的正方形结构元素腐蚀后，由于上部的十字形物体无法完全包含结构元素，故而完全消失；底部的第一个正方形同结构元素大小形状完全一致，刚好能够包含结构元素，所以经过腐蚀留下了正中心的1个像素；其余物体边界均“瘦”了1个像素。

为了比较效果，图10.6（c）中换用了3×3的十字结构元素，这时图10.6（b）中消失的顶部的十字形物体因为可以正好包含结构元素了而被保留了1个中心像素，还需要注意的是连通条带连接端和图10.6（b）中的区别。

图10.6（d）中采用5×5的正方形结构元素后，所有物体边界继续缩小，而小于它的物体都完全消失，原始图像中5个像素宽的连通条带刚好还剩下中间的1个像素宽的连通线。

当图10.6（e）中正方形结构元素增大到6×6之后，原本5个像素宽的连通条带再也无法包含整个结构元素，从而在腐蚀后的图像中彻底消失，而底部也只有原本 6×6 的正方形物体还仅存 1个像素。

采用7×7的正方形结构元素腐蚀之后，图10.6（f）中只保留了两个主要的物体，而圆形物体和矩形物体的边界也都分别缩减了3个像素。

例 10.1 中，随着腐蚀结构元素的逐步增大，小于结构元素的物体相继消失。由于腐蚀运算这样的特点，可以用于滤波。选择适当大小和形状的结构元素，可以滤除掉所有不能完全包含结构元素的噪声点。然而，利用腐蚀滤除噪声有一个缺点，即在去除噪声点的同时，对图像中前景物体的形状也会有影响，但当我们只关心物体的位置或者个数时，这不会有什么问题。

10.2.2 膨胀及其实现

1．理论基础

对Z^2上元素的集合A和S，使用S对A进行膨胀，记作A⊕S，形式化地定义为

$$A \oplus S = \{ z \mid (\hat{S})_z \cap A \neq \phi \}; \qquad (10\text{-}2)$$

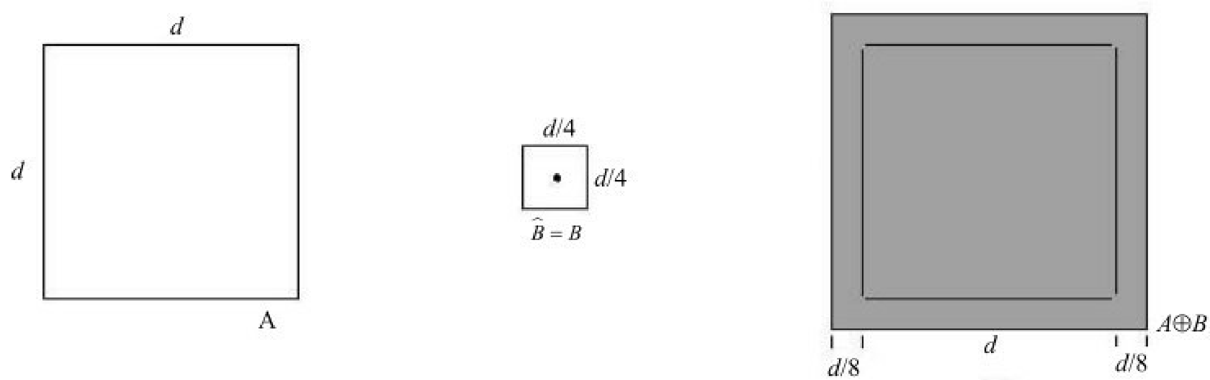
设想有原本位于图像原点的结构元素S，让S在整个Z^2平面上移动，

当其自身原点平移至z点时S相对于其自身的原点的映像$\hat{S}$和A有公共的交集，即$\hat{S}$和A至少有1个像素是重叠的，则所有这样的z点构成的集合为S对A的膨胀图像，如图10.7所示。

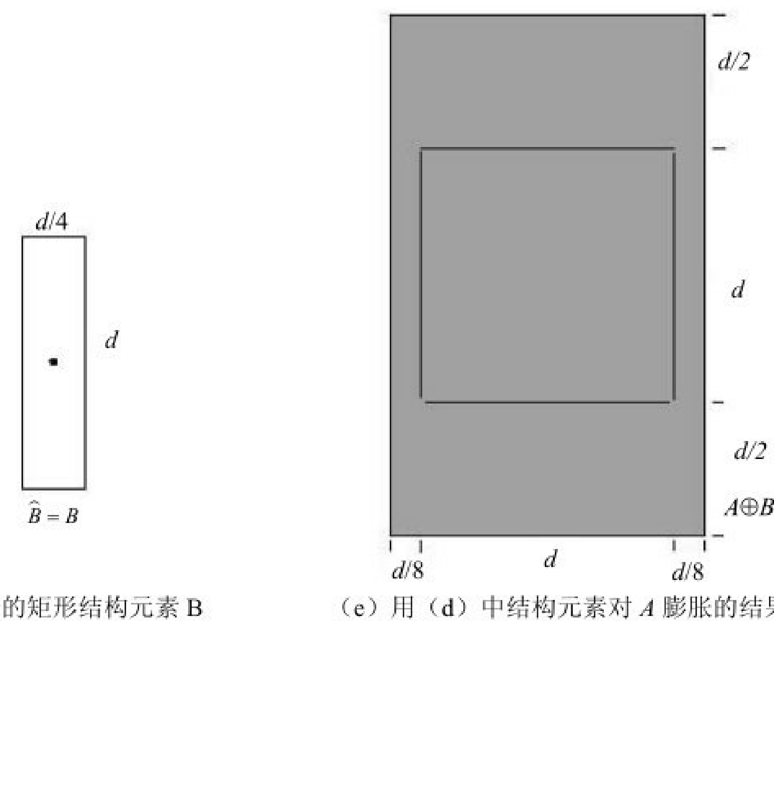
实际上，膨胀和腐蚀对于集合求补和反射运算是彼此对偶①的，即

【注】① 注：对偶的意义：某个图像处理系统用硬件实现了腐蚀运算，那么不必再另搞一套膨胀的硬件，直接利用该对偶就可以实现了。

$$(A \ominus B)^c = A^c \oplus \hat{B} \tag{10-3}$$



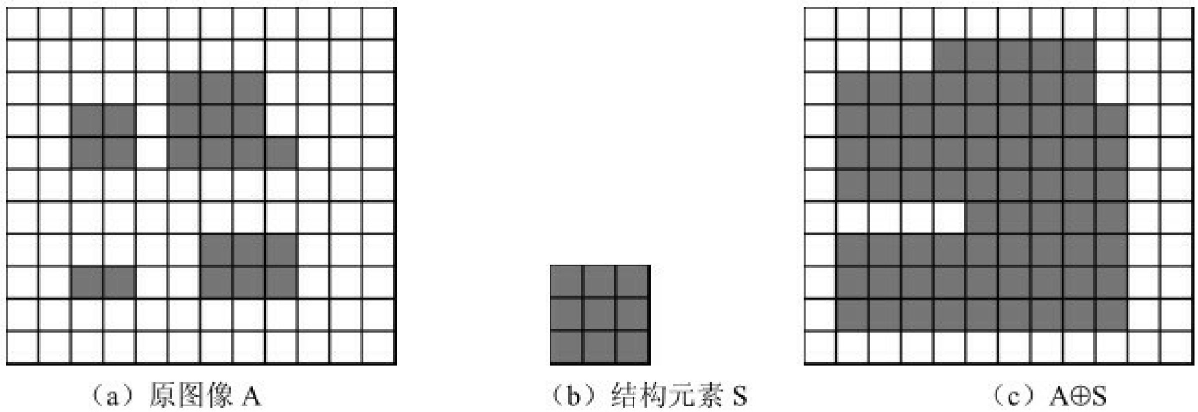
（a）集合 *A*　（b）正方形结构元素 B（黑点是原点）　（c）*B* 对 *A* 膨胀后的结果



（d）拉长的矩形结构元素 B　（e）用（d）中结构元素对 *A* 膨胀的结果

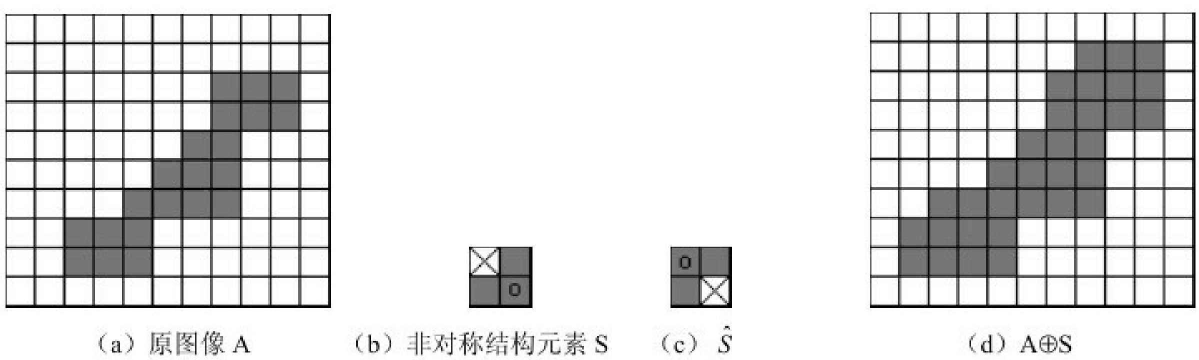
▲图10.7 膨胀示意图（图中标出了所有图形的长和宽）

采用原点位于中心的3×3对称结构元素的膨胀运算效果模拟如图10.8所示，物体间小于3个像素的缝隙都被膨胀所弥合。



▲图10.8 膨胀运算模拟

这里值得注意的是定义中要求和A有公共交集的不是结构元素S本身，而是S的反射集$\hat{S}$，觉得熟悉吗？这种形式似乎容易让我们回忆起卷积运算，而腐蚀在形式上则更像相关运算。由于图10.8中使用的是对称的结构元素，故使用S和$\hat{S}$的膨胀结果相同。但对于图10.9所示的非对称结构元素的膨胀示例，则会产生完全不同的结果，因此在实现膨胀运算时一定要先计算$\hat{S}$ 。



▲图10.9 非对称结构元素的膨胀运算

2．Matlab实现

imdilate函数用于完成图像膨胀，其常用调用形式如下。

I2 = imdilate(I, SE);

参数说明

● I为原始图像，可以是二值或灰度图像（对应于灰度膨胀）。

● SE是由strel函数返回的自定义或预设的结构元素对象。

返回值

● I2为膨胀后的输出图像。

3．膨胀的作用

和腐蚀相反，膨胀能使物体边界扩大，具体的膨胀结果与图像本身和结构元素的形状有关，膨胀常用于将图像中原本断裂开来的同一物体桥接起来，对图像进行二值化之后，很容易使得一个连通的物体断裂为两个部分，而这会给后续的图像分析(如要基于连通区域的分析统计物体的个数)造成困扰，此时就可借助膨胀桥接断裂的缝隙。

[例10.2]形态学腐蚀和膨胀的应用——文字断裂与桥接。

相应的程序如下。

```
>> I = imread('../starcraft.bmp');    %读入图像
>> figure, imshow(I); %得到图10.10（a）
>> Ie1 = imerode(I, [1 1 1; 1 1 1; 1 1 1]) ;    %3×3正方形结构元素的腐蚀
>> figure, imshow(Ie1); %得到图10.10（b）
>> Ie2 = imerode(Ie1, [0 1 0; 1 1 1; 0 1 0]); %3×3十字形结构元素的腐蚀
>> figure, imshow(Ie2); %得到图10.10（c）
>> Id1 = imdilate(Ie2, [1 1 1; 1 1 1; 1 1 1]);    %3×3正方形结构元素的
```

膨胀

```
>> figure, imshow(Id1); %得到图10.10（d）
>> Id2 = imdilate(Id1, [1 1 1; 1 1 1; 1 1 1]);  %3×3正方形结构元素的膨胀
>> figure, imshow(Id2); %得到图10.10（e）
>> Id3 = imdilate(Id2, [0 1 0; 1 1 1; 0 1 0]);
>> figure, imshow(Id3); %得到图10.10（f）  %3×3十字形结构元素的膨胀
```

上述程序的运行结果如图10.10所示。



（a）原始图像 starcraft.bmp　（b）（a）经过 3×3 正方形结构元素腐蚀

（c）（b）经过 3×3 十字形结构元素腐蚀　（d）（c）经过 3×3 正方形结构元素膨胀

（e）（d）经 3×3 正方形结构元素膨胀　（f）（e）经 3×3 十字形结构元素膨胀

▲图10.10 膨胀示例

10.2.3 开运算及其实现

开运算和闭运算都由腐蚀和膨胀复合而成，开运算是先腐蚀后膨胀；而闭运算是先膨胀后腐蚀。下面先介绍开运算，10.2.4小节介绍比

运算。

1．理论基础

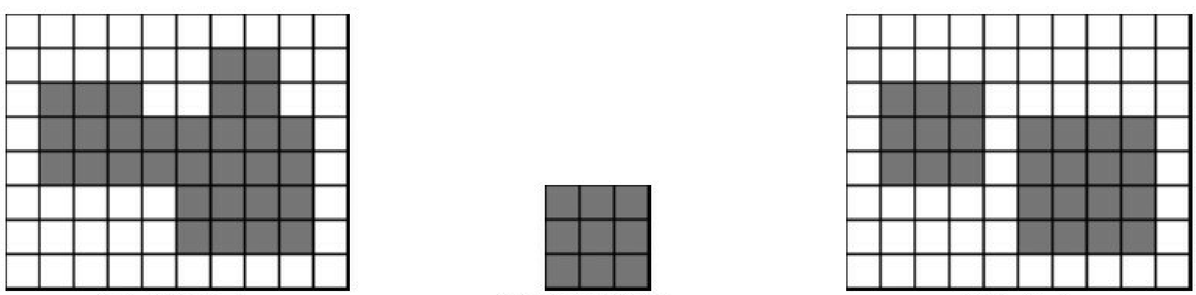
使用结构元素S对A进行开运算，记作A∘S，可表示为

$$A \circ S = (A \ominus S) \oplus S \tag{10-4}$$

一般来说，开运算使图像的轮廓变得光滑，断开狭窄的连接和消除细毛刺。

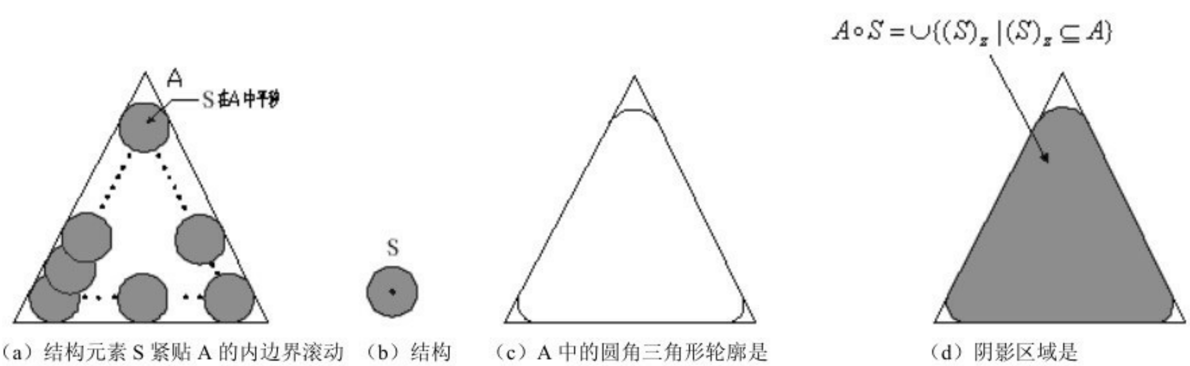
如图 10.11 所示，开运算断开了图中两个小区域间两个像素宽的连接（断开了狭窄连接），并且去除了右侧物体上部突出的一个小于结构元素的2×2的区域（去除细小毛刺）；但与腐蚀不同的是，图像大的轮廓并没有发生整体的收缩，物体位置也没有发生任何变化。

根据图10.12所示的开运算示意图，可以帮助您更好地理解开运算的特点。为了比较，图10.12中也标示出了相应的腐蚀运算的结果。



（a）原图像 A　（b）结构元素 S　（c）$A \circ S$

▲图10.11 开运算效果模拟



（a）结构元素 S 紧贴 A 的内边界滚动　（b）结构元素 S　（c）A 中的圆角三角形轮廓是开运算的外部边界　（d）阴影区域是开运算的结果

▲图10.12 开运算示意图

在图10.12中，让结构元素S紧贴A的内边界滚动，滚动过程中始终保证S完全包含于A，此时S中的点所能达到的最靠近A的内边界的位置就构成了图10.12（c）所示的开运算的外边界。从这个意义上开运算可以表示为$A \circ S = \cup \{(S)_z | (S)_z \subseteq A\}$；而此时S的中心所能达到的最靠近A的内边界的位置就构成了S对A的腐蚀的外边界[图10.12（a）中的虚线轮廓]。

2．Matlab实现

根据定义，以相同的结构元素先后调用imerode和imdilate即可实现开操作。此外，Matlab中也直接提供了开运算函数imopen，其调用形式如下。

I2 = imopen(I, SE);

参数说明

● I为原始图像，可以是二值或灰度图像（对应于灰度开）。

● SE是由strel函数返回的自定义或预设的结构元素对象。

返回值

● I2为开运算后的输出图像。

利用imopen对例10.1中的图像erode_dilate.bmp进行开运算的Matlab程序如下。

```
>> I = imread('erode_dilate.bmp');
>> figure, imshow(I, []); % 显示图10.13（a）
>> Io = imopen(I, ones(6, 6)); % 采用6*6的正方形结构元素开运算
>> figure, imshow(Io, []); % 显示图10.13（b）
```

上述程序运行后效果如图10.13所示。

从图10.13中可以看到同腐蚀相比，开运算在过滤噪声的同时并没有对物体的形状、轮廓造成明显的影响，这是一大优势。但当我们只关

心物体的位置或者个数时，物体形状的改变不会给我们带来困扰，此时用腐蚀滤波具有处理速度上的优势（同开运算相比节省了一次膨胀运算）。



（a）erode_dilate.bmp　　（b）图（a）的开运算结果

▲图10.13 开运算结果

10.2.4 闭运算及其实现

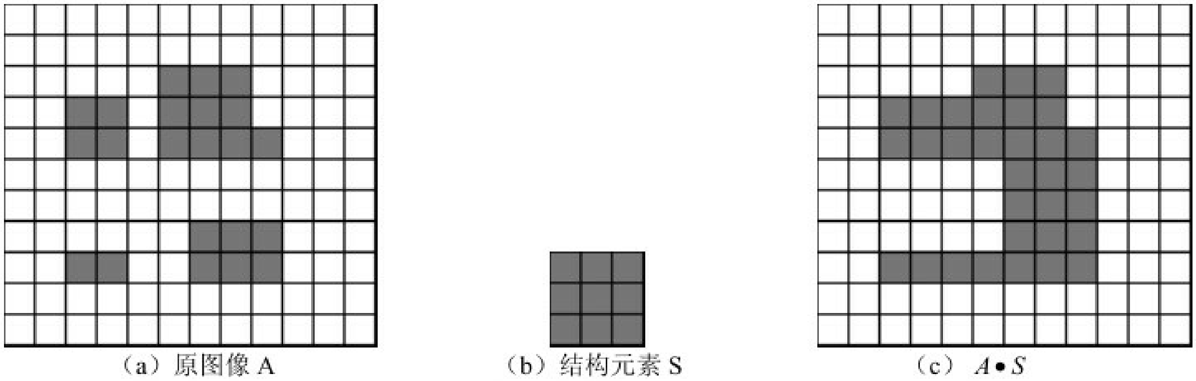
1．理论基础

使用结构元素S对A进行闭运算，记作A•S，可表示为

$$A \bullet S = (A \oplus S) \ominus S \qquad (10\text{-}5)$$

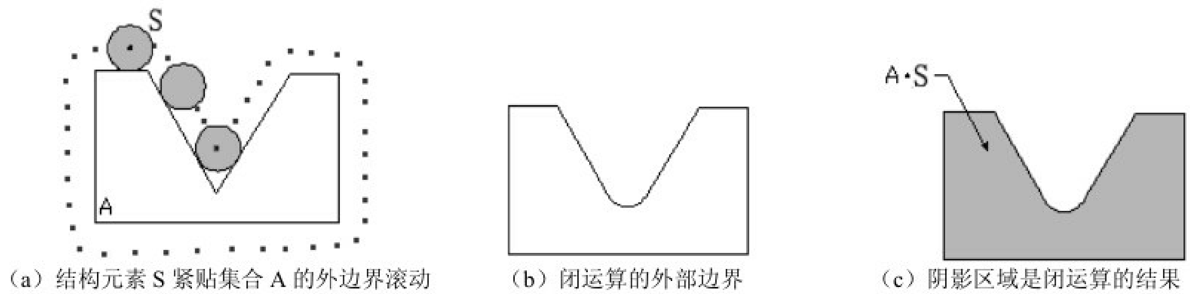
闭运算同样使轮廓变得光滑，但与开运算相反，它通常能够弥合狭窄的间断，填充小的孔洞。

与前面图10.8膨胀运算效果不同，图10.14所示的闭运算在前景物体整体位置和轮廓不变的情况下，弥合了物体之间宽度小于3个像素的缝隙。



▲图10.14 闭运算效果模拟

根据图10.15所示的闭运算示意图，可以帮助您更好地理解闭运算的特点，为了比较，图10.15中也给出了相应的膨胀运算的结果。



▲图10.15 闭运算示意图

这次我们让结构元素S紧贴A的外边界滚动，滚动过程中始终保证S不完全离开A （ $(S)_z \cap A \neq \phi$ ），此时S中的点所能达到的最靠近A的外边界的位置就构成了图10.12（c）所示的闭运算的外边界。而此时S中心点所能达到的最靠近A的内边界的位置就构成了S对A的膨胀的外边界（图10.15（a）中的虚线轮廓）。

从图10.12和图10.15我们还注意到，在圆形的结构元素作用下，开运算使得物体小于180°的拐角变得圆滑，大于180°的角则没有变化，腐蚀运算则刚好与开运算相反；而闭运算使得物体大于180°的拐角变得圆滑，而小于180°的拐角则没有变化，膨胀运算则刚好与闭运算相反。

最后还需要说明的是开闭运算也是对偶的，然而与腐蚀和膨胀不同的是，对于某图像多次应用开或闭运算和只进行一次运算的效果相同，即有

$$(A \circ B) \circ B = A \circ B, \quad (A \bullet B) \bullet B = A \bullet B \qquad (10\text{-}6)$$

2．Matlab实现

根据定义，以相同的结构元素先后调用imdilate和imerode即可实现闭操作。此外，Matlab中也直接提供了闭运算函数imclose，其用法同imopen类似，这里不再赘述。

10.3 二值图像中的形态学应用

本节将介绍一些非常经典的形态学应用，它们都是通过将 10.2 节中的基本运算按照特定次序组合起来，并且采用一些特殊的结构元素而实现的。

10.3.1小节将要介绍的击中击不中变换主要用于图像中某种特定形状的精确定位，其后将要讨论的很多形态学应用，如轮廓、细化以及像素化等都有助于我们抓住物体最本质的特征（轮廓、形状，或位置），这些特征都是强有力的图像描绘子，经处理后可作为后续的识别任务所需要的特征。

10.3.1 击中与击不中变换及其实现

1．理论基础

形态学击中击不中变换常用于图像中某种特定形状的精确定位，是一种形状检测的基本工具。记作A ⊛ S，可表示为

$$A \circledast S = (A \ominus S1) \cap (A \ominus S2) \qquad (10\text{-}7)$$

其中， $S1\cap S2=\varphi$且$S=S1\cup S2$，实际上S1代表S中我们感兴趣的物体（要检测的形状）对应的集合，而S2为S中背景部分对应的集合。

从式（10-7）上分析，击中与击不中变换首先用我们感兴趣的物体S1去腐蚀图像A，得到的结果是使S1完全包含于A中前景部分时其中心点位置的集合U1，可以将U1看作是S1在A中所有匹配的中心点的集合。为了在A中精确地定位S1而排除掉那些仅仅包含S1但不同于S1的物体或区域，有必要引入和S1相关的背景部分S2，一般来说S2是在S1周围包络着S1的背景部分， S1和S2和在一起组成了S，式（10-7）中的后一半正是计算图像A的背景A^c和S的背景部分S2的腐蚀，得到的结果U2是使S的背景部分S2完全包含于A^c时S中心位置的集合。U1和U2的交集自然就是这样一些点p的集合：当S中心位于p时，S的前景（物体）部分S1和A中的某个前景部分完全重合，而S的背景部分也和A的某个背景部分完全重合，而S1又是包络在S2其中的，从而保证了我们感兴趣的物体S1在图像A的p点处找到了一个精确匹配。读者可再结合例10.3体会击中击不中变换的原理。

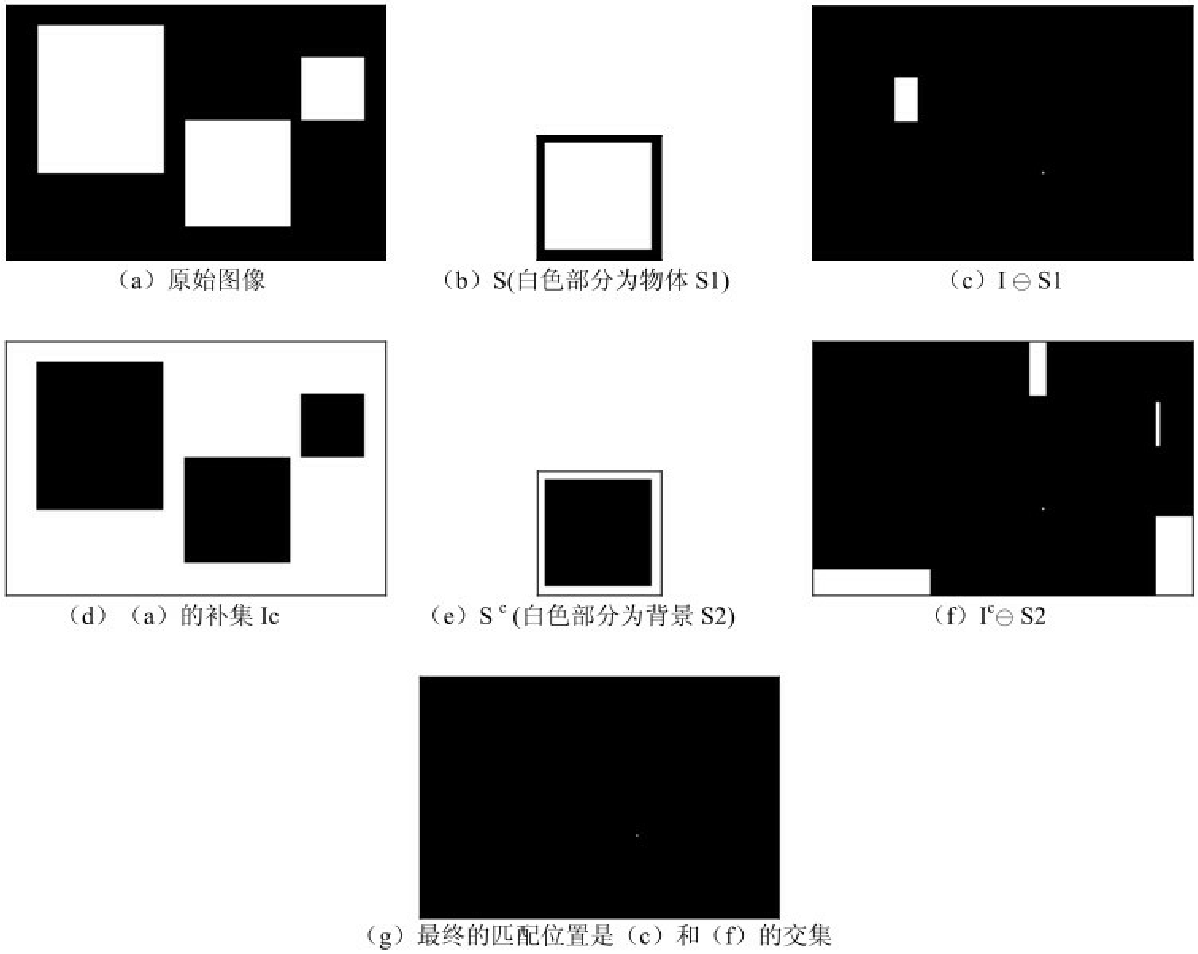
[例10.3]击中击不中变换。

下面的程序首先生成了一个如图10.16（a）所示的原始图像，其中最左侧为一个70×60的矩形物体X，居中靠下的为一个50×50的正方形Y，右上方的正方形Z的边长为30；而后生成了如图10.16（b）所示的结构元素S和物体S1；接下来，程序根据式（10-7）计算I与S的击中击不中变换$I \circledast S$。

```
>> % 生成原始图像
>> I = zeros(120, 180);
>> I(11:80, 16:75) = 1;
>> I(56:105, 86:135) = 1;
>> I(26:55, 141:170) = 1;
>> figure, imshow(I);    %得到图10.16（a）
```

```
>>
>> % 生成结构元素S
>> se = zeros(58, 58);
>> se(5:54, 5:54) = 1;    % 物体S1
>> figure, imshow(se);    %得到图10.16（b）
>>
>> % 击中击不中变换
>> Ie1 = imerode(I, se);    % 物体腐蚀
>> figure, imshow(Ie1);    %得到图10.16（c）
>> Ic = 1-I;    % I的补
>> figure, imshow(Ic);    %得到图10.16（d）
>> S2 = 1-se;
>> figure, imshow(S2);    %得到图10.16（e）
>> Ie2 = imerode(Ic, S2);    % 背景腐蚀 %得到图10.16（f）
>> Ihm = Ie1 & Ie2;    % 两次腐蚀的交集
>> figure, imshow(Ihm);    %得到图10.16（g）
```

上述程序的运行结果如图10.16所示。



▲图10.16 击中击不中变换

图10.16（g）给出了变换的最终结果。为便于观察在显示时每幅图像周围都环绕着一圈黑色边框，注意该边框并不是图像本身的一部分。

注意

对于结构元素S，我们感兴趣的物体S1之外的背景S2不能选择得太宽，因为使得S包含背景S2的目的仅仅是定义出物体S1的外轮廓，以便在图像中能够找到准确的完全匹配位置。从这个意义上说，物体S1周围有一个像素宽的背景环绕就足够了，例10.3中选择了4个像素宽的背景，是为了使结构元素背景部分S2看起来比较明显，但如果背景部分过大，则会影响击中击不中变换的计算结果，在上例中中间的正方形Y与

右上的正方形Z之间的水平距离为6，如果在定义S时，S2的宽度超过6个像素，则最终的计算结果将是空集。

根据式（10-3）给出的对偶关系，式（10-7）还可表示为如下两种形式，然而，式（10-7）更为直观，更易理解。

$$A \circledast S = (A \ominus S1) \cap (A \oplus \hat{S}_2)$$

$$= (A \ominus S1) - (A \oplus \hat{S}_2) \quad (10\text{-}8)$$

2．Matlab实现

IPT中进行击中击不中变换的函数为bwhitmiss，调用形式为

Ihm = bwhitmiss(I, S1, S2);

参数说明

● I为输入图像。

● S1和S2即为式（10-7）中介绍过的结构元素。

返回值

● Ihm是击中击不中变换后的结果图像。

如果使用bwhitmiss函数来完成例10.3，则只需下面的一句代码。

Ihm = bwhitmiss(I, se, S2); %se和S2与在例10. 3中时的意义相同

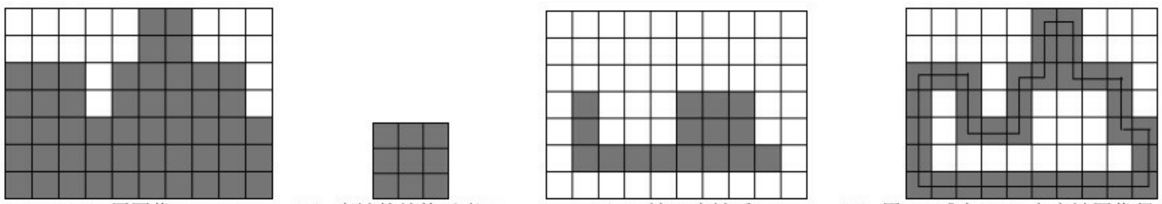
10.3.2 边界提取与跟踪及其实现

轮廓是对物体形状的有力描述，对于图像分析和识别十分有用。通过边界提取算法可以得到物体的边界轮廓；而边界跟踪算法在提取边界的同时还能依次记录下边界像素的位置信息，下面分别介绍。

1．边界提取

要在二值图像中提取物体的边界，容易想到的一个方法是将所有物体内部的点删除（置为背景色）。具体地说，可以逐行扫描原图像。如果发现一个黑点（图10.17中黑点为前景点）的8个邻域都是黑点，则该

点为内部点，在目标图像中将它删除。实际上这相当于采用一个3×3的结构元素对原图像进行腐蚀，使得只有那些8个邻域都有黑点的内部点被保留，再用原图像减去腐蚀后的图像，恰好删除了这些内部点，留下了边界像素。这一过程可参见图10.17。



（a）原图像 A　（b）腐蚀的结构元素 S　（c）A 被 S 腐蚀后　（d）用 A 减去（c）中腐蚀图像得到的边界图像，在图中贯穿像素中心的一条黑色线标出了 1 个像素宽的边界

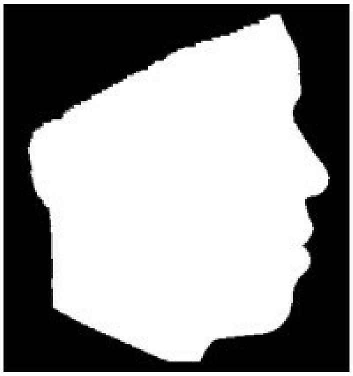
▲图10.17 边界提取

[例10.4]边界轮廓提取。

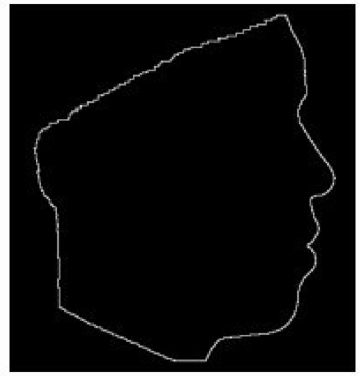
采用前述方法提取提取二值人脸图像边界轮廓的Matlab程序如下：

```
>> I = imread('../head_portrait.bmp'); %读入原图像
>> figure, imshow(I);    %得到图10.18（a）中的图像
>> se = strel('square', 3);    %3*3的正方形结构元素
>> Ie = imerode(I, se);    %腐蚀得到内部点
>> Iout = I - Ie;    %减去内部点留下边界点
>> figure, imshow(Iout);    %得到图10.18（b）中的图像
```

上述程序的运行结果如图10.18所示。



（a）侧面轮廓的二值化图像 head_portrait.bmp



（b）采用上述方法边界提取后图像

▲图10.18

2．边界跟踪

为了依次记录下边界上的各个像素，边界跟踪首先按照某种扫描规则找到目标物体边界上的一个像素，而后就以该像素为起始点，根据某种顺序（如顺时针或逆时针）依次找出物体边界上的其余像素，直到又回到了起始点，完成整条边界的跟踪。

例如，我们可以按照从左到右、从上到下的顺序扫描图像，这样首先会找到目标物体最左上方的边界点P0。显然，这个点的左侧以及上侧都不可能存在边界点（否则左侧或者上侧的边界点就会成为第一个被扫描到的边界点），因此不妨从左下方向逆时针开始探查，如左下方的点是黑点，直接跟踪至此边界点，否则探查方向逆时针旋转45°，直至找到第一个黑点为止，跟踪至此边界点。找到边界点后，在当前探查方向的基础上顺时针回转90°，继续用上述方法搜索下一个边界点，直到探查又回到初始的边界点P0，完成了整条边界的跟踪。整个跟踪过程如图10.19所示。

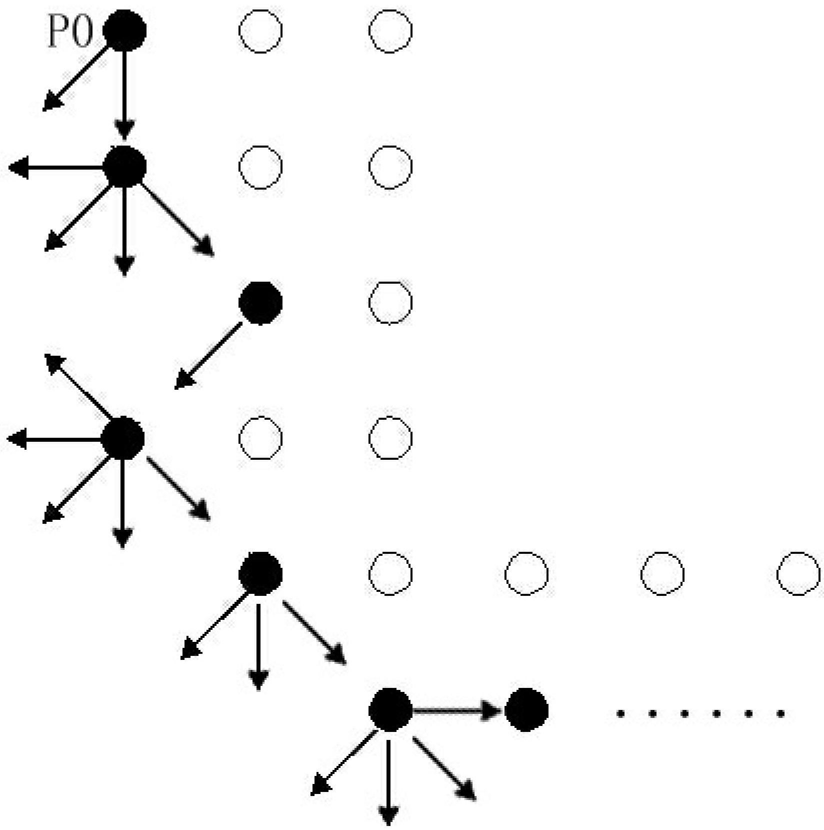


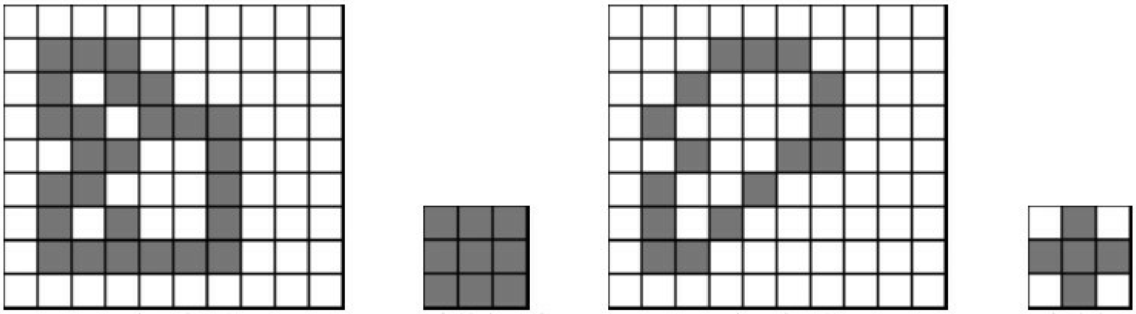
图 10.19 轮廓跟踪算法示意

10.3.3 区域填充

区域填充可视为边界提取的反过程，它是在边界已知的情况下得到边界包围的整个区域的形态学技术。

问题的描述如下。已知某一8连通边界和边界内部的某个点，要求从该点开始填充整个边界包围的区域，这一过程也称为“种子”填充，填充的开始点被称为“种子”。

首先我们注意到，对于4连通的边界，其围成的内部区域是8连通的，而8连通的边界围成的内部区域却是4连通的，如图10.20所示。



（a）4 连通边界的 8 连通内部区域　（b）用来填充 4 连通边界的 3×3 结构元素　（c）8 连通边界的 4 连通内部区域　（d）用来填充 8 连通边界的十字结构元素

▲图10.20

为了填充4连通的边界应选择图10.20（b）中的3×3结构元素，而为在8连通边界内从种子点得到区域我们选用图10.20（d）的十字结构元素S对初始时仅为种子点的图像B进行膨胀，十字结构元素S能够保证只要B在边界A的内部（不包括边界本身），每次膨胀都不会产生边界之外的点（新膨胀出来的点要么在边界内部，要么最多落在边界上），这样只需把每次膨胀的结果图像和边界的补图像A^c 相交，就能把膨胀限制在边界内部。随着对B的不断膨胀，B的区域不断生长，但每次膨胀后与A^c 的交又将B限制在边界A的内部，这样一直到最终B充满整个A的内部区域，停止生长。此时的B与A的并集即为最终的区域填充结果。

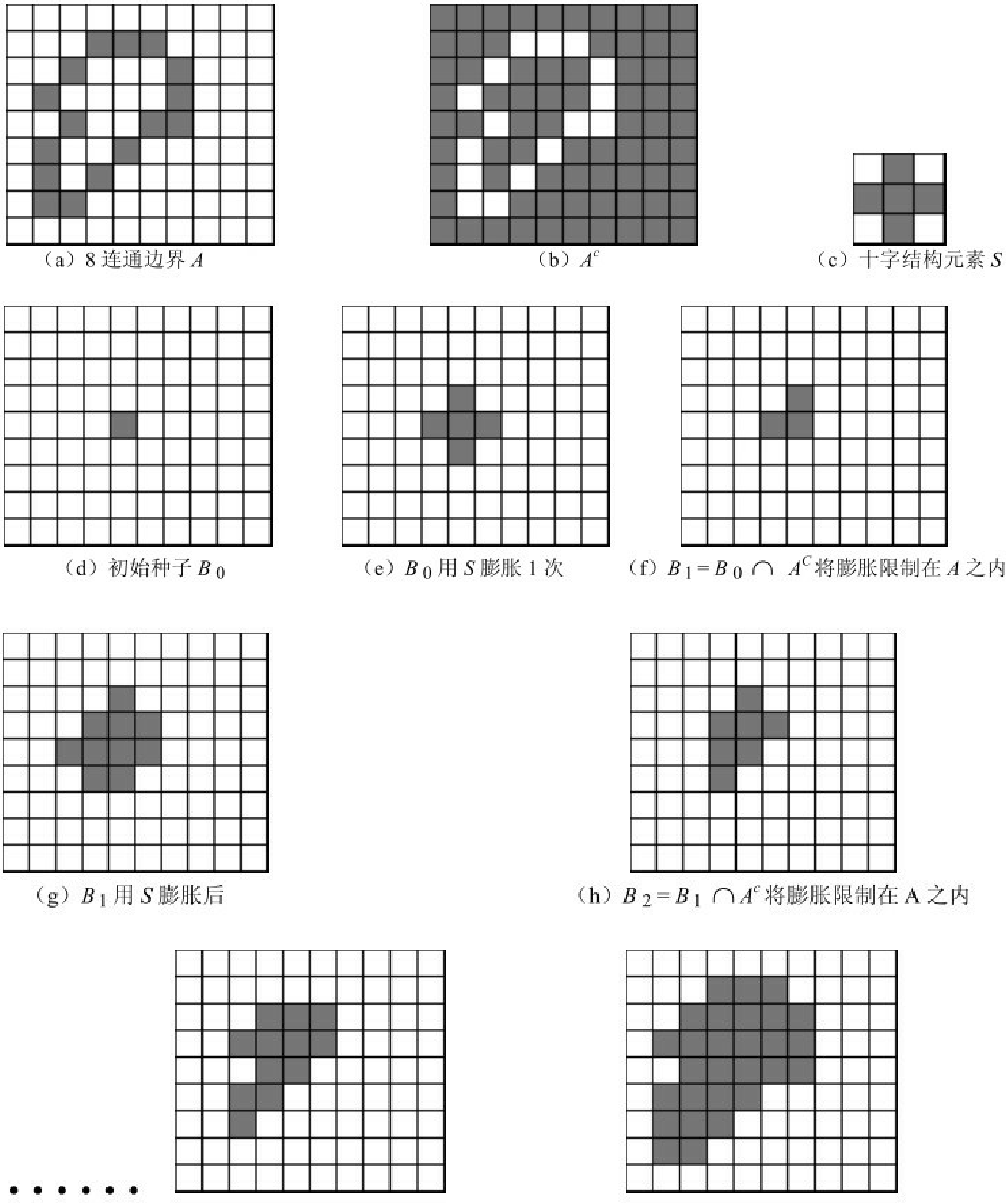
算法概要如下。

初始化：b_0=种子点

循环： Do $b_{i+1}=(b_i \oplus S) \cap A^c$

Until $b_{i+1}==b_i$

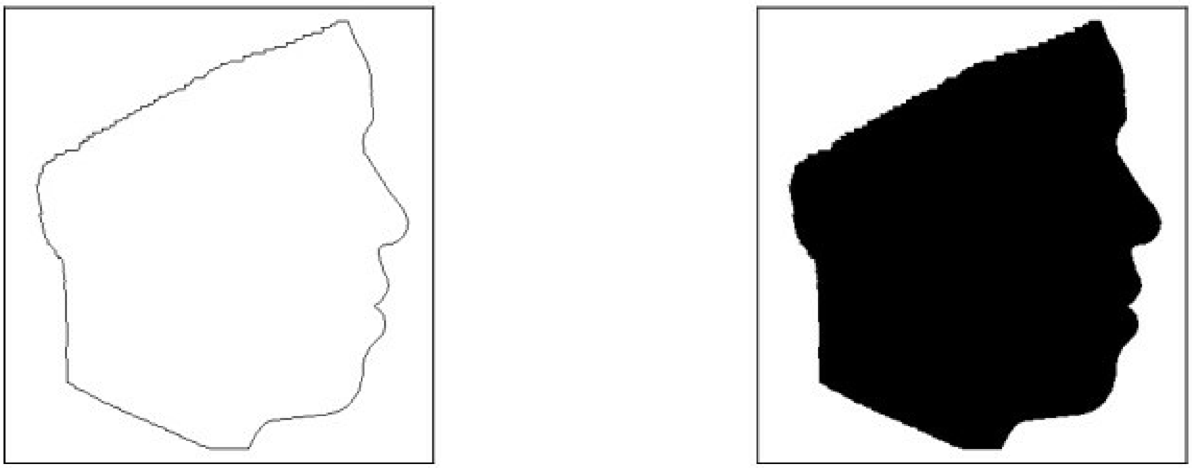
图10.21（d）～（j）形象地模拟了整个区域填充过程。



▲图10.21 区域填充效果模拟

在 Matlab 中，可以使用bwmorph函数方便地实现区域填充。关于

bwmorph函数将在 10.3.8小节中介绍。一个区域填充的实际效果如图10.22所示。



（a）原图像　　（b）图（a）的填充效果

▲图10.22 区域种子填充效果（种子点设置于图像中心）

10.3.4 连通分量提取及其实现

连通分量的概念在1.3.1小节中曾介绍过。在二值图像中提取连通分量是许多自动图像分析应用中的核心任务。提取连通分量的过程实际上也是标注连通分量的过程，通常的做法是给原图像中的每个连通区分配一个唯一代表该区域的编号，在输出图像中该连通区内的所有像素的像素值就赋值为该区域的编号，我们将这样的输出图像称为标注图像。

1．理论基础

这里准备介绍一种是基于形态学的膨胀操作的提取连通分量的方法，另一种递归的方法将在10.3.6小节的像素化算法中给出。以8连通的情况为例，对于图10.23（a）的内含多个连通分量的图像A，从仅为连通分量A1内部的某个点的图像B开始，不断采用如图10.23（c）所示的结构元素S进行膨胀。由于其他连通分量与A1之间至少有一条1个像素宽的空白缝隙（图10.23（a）中的虚线），3×3的结构元素保证了只要B

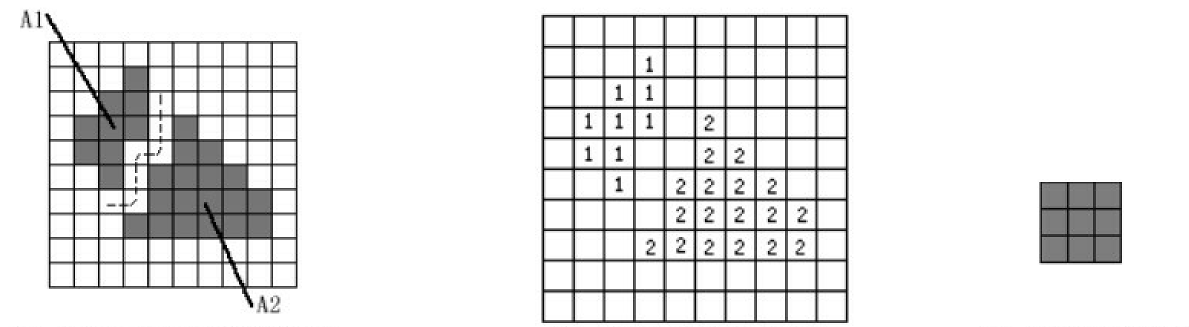
在区域A1的内部，则每次膨胀都不会产生位于A中其他连通区域之内的点，这样只需用每次膨胀后的结果图像和原始图像A相交，就能把膨胀限制在A1内部。随着对B的不断膨胀，B的区域不断生长，但每次膨胀后与A的交又将B限制在连通分量A1的内部，直到最终B充满整个连通分量A1，对连通分量A1的提取完毕。

算法概要如下。

初始化：b_0=连通分量A1中的某个点

循环： Do $b_{i+1}=(b_i \oplus S)\cap A$

Until $b_{i+1}==b_i$



（a）含有 2 个连通分量的图像 A　（b）A 的连通分量标注图像　（c）3×3 结构元素 S

▲图10.23 连通分量提取结果

提取连通分量的算法与区域填充算法十分相似，只需改变膨胀结构元素（8连通使用3×3的正方形结构元素，4连通使用3×3十字形结构元素）并且把每次膨胀后同A^c 的交改为同A的交。

2．Matlab实现

在Matlab中，连通分量的相关操作主要借助IPT函数bwlabel实现，其调用语法为

[L num] = bwlabel(Ibw, conn)

参数说明

● Ibw为一幅输入二值图像。

● conn为可选参数，指明要提取的连通分量是4连通还是8连通，默认值为8。

返回值

● L为类似于图10.23（b）的标注图像。

● num为二值图像Ibw中连通分量的个数。

提取连通分量的应用十分广泛，利用标注图像可以方便地进行很多基于连通区的操作。例如要计算某一连通分量的大小，只需扫描一遍标注图像，对像素值为该区编号的像素进行计数；又如要计算某一连通分量的质心，只需扫描一遍标注图像，找出所有像素值为该区编号的像素的x、y坐标，然后计算其平均值。

下面结合2个Matlab实例来说明。

[例10.5]在人脸局部图像中定位嘴的中心。

我们希望在如图10.24（a）所示的图像中定位嘴的中心，假定已经掌握了输入图像中的某些经验知识，即嘴部占据了图像的大部分区域且从灰度上易于与周围皮肤分离开来。于是针对性地拟定了在二值化图像中寻找最大连通区域中心的解决方案，具体步骤如下。

（1）对输入图像进行二值化处理。

（2）标注二值图像中的连通分量。

（3）找出最大的连通分量。

（4）计算最大连通分量的中心。

依照上述思路实现的 Matlab 代码如下，读者可以在金羽图书论坛（http://bbs.book95.com）的“金羽图书与答疑”板块与本书同名的主题帖子附件的第10章Code目录下的locateMouth.m文件中找到。

```
% locateMouth.m
I = imread('../mouth.bmp'); %读入图像
Id = im2double(I);
figure, imshow(Id) % 得到10.24（a）
```

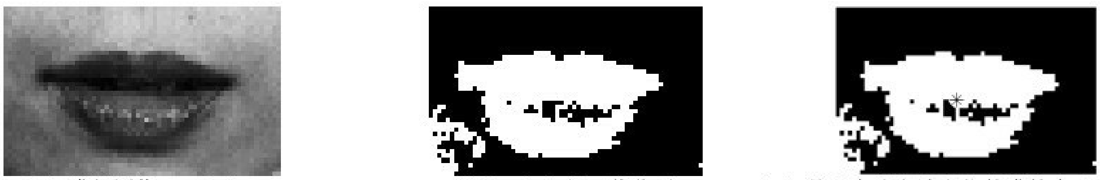
```
Ibw = im2bw(Id, 0.38); %以0.38为阈值二值化
Ibw = 1 - Ibw; %为在Matlab中进行处理，将图像反色
figure, imshow(Ibw) % 得到10.24（b）
hold on
[L, num] = bwlabel(Ibw, 8); % 标注连通分量
disp(['图中共有' num2str(num) '个连通分量'])
% 找出最大的连通分量（嘴）
max = 0; % 当前最大连通分量的大小
indMax = 0; % 当前最大连通分量的索引
for k = 1:num
  [y x] = find(L == k); % 找出编号为k的连通区的行索引集合y和列
索引集合x
  nSize = length(y); % 计算该连通区的像素数目
  if(nSize > max)
    max = nSize;
    indMax = k;
  end
end
if indMax == 0
  disp('没有找到连通分量')
  return
end
% 计算并显示最大连通分量（嘴）的中心
[y x] = find(L == indMax);
●Mean = mean(y);
xMean = mean(x);
plot(xMean, yMean, 'Marker', 'o', 'MarkerSize', 14, 'MarkerEdgeColor',
```

'w',

'MarkerFaceColor', 'w');

plot(xMean, yMean, 'Marker', '*', 'MarkerSize', 12, 'MarkerEdgeColor', 'k'); %得到10.24（c）

上述程序运行后结果如图10.24所示。



（a）嘴部图像 mouth.bmp （b）（a）经二值化后 （c）基于上述方法定位的嘴的中心，用*标出

▲图10.24 嘴部定位

我们看到在以0.38为阈值分割后，得到的图10.24（b）中的二值图像共有4个连通分量，结合经验知识，可以认为最大的连通分量对应于嘴部。语句[y x] = find(L = = k)用来在标注图像L中找出编号为k的连通区的行索引集合y和列索引集合x，找出所有连通区中最大的一个，计算向量y和x的平均值，即为该区域中心的坐标，结算结果在图10.24（c）中用*显示了出来。

[例10.6]细菌计数。

要对图10.25（a）中显微镜视野内的细菌进行计数，我们的思路是在二值化后的黑白图像中统计连通区的个数，从而确定细菌的个数。首先对原始图像进行阈值处理，得到图10.25（b）中的二值化图像，注意到下方的一个细菌出现了“断裂”，这可能是由于阈值选择不当（这里是阈值偏高）或是图像整体灰度不均造成根本不存在一个能够正确分割出所有物体的阈值。

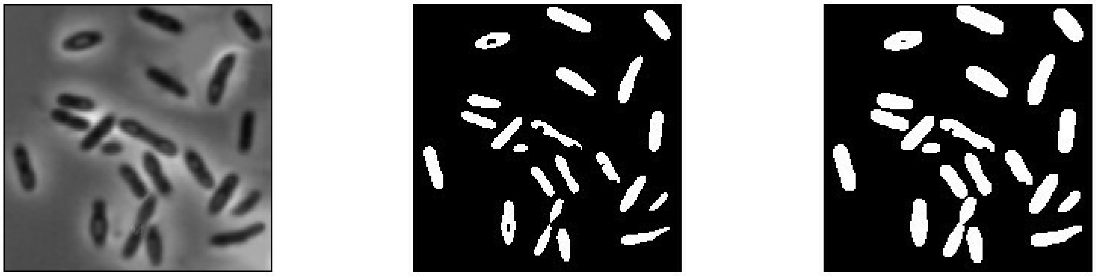
事实上，“断裂”和“合并”都会给计数带来困扰，针对图10.25（b）阈值偏高易产生“断裂”的特点，我们对图10.25（b）中二值图像采用

3×3的结构元素进行膨胀，在膨胀后的图像图10.25（c）中，我们看到“断裂”被成功“接合”，同时又没有产生不同细菌的“合并”，再统计图10.25（c）中的二值图像中连通分量的数目，即得到细菌的准确计数。

相应的Matlab代码如下。

```
>> I = imread('../bw_bacteria.bmp'); % 读入二值化后的细菌图像
>> figure, imshow(I) % 得到（b）图
>> [L, num] = bwlabel(I, 8); % 直接统计（b）中连通区个数
>> num % 显示细菌个数，由于“断裂”存在，比实际数目多1
num =
   22
>> Idil = imdilate(I, ones(3,3));    % 采用3×3的结构元素膨胀
>> figure, imshow(Idil)    % 得到（c）图
>> [L, num] = bwlabel(Idil, 8);    % 统计（c）中的连通区个数
>> num % 实际的细菌个数
num =
   21
```

上述程序运行后的效果如图10.25所示。



（a）显微镜下的细菌图像　（b）将（a）中图像二值化后　（c）对（b）中图像采用3×3的结构元素膨胀

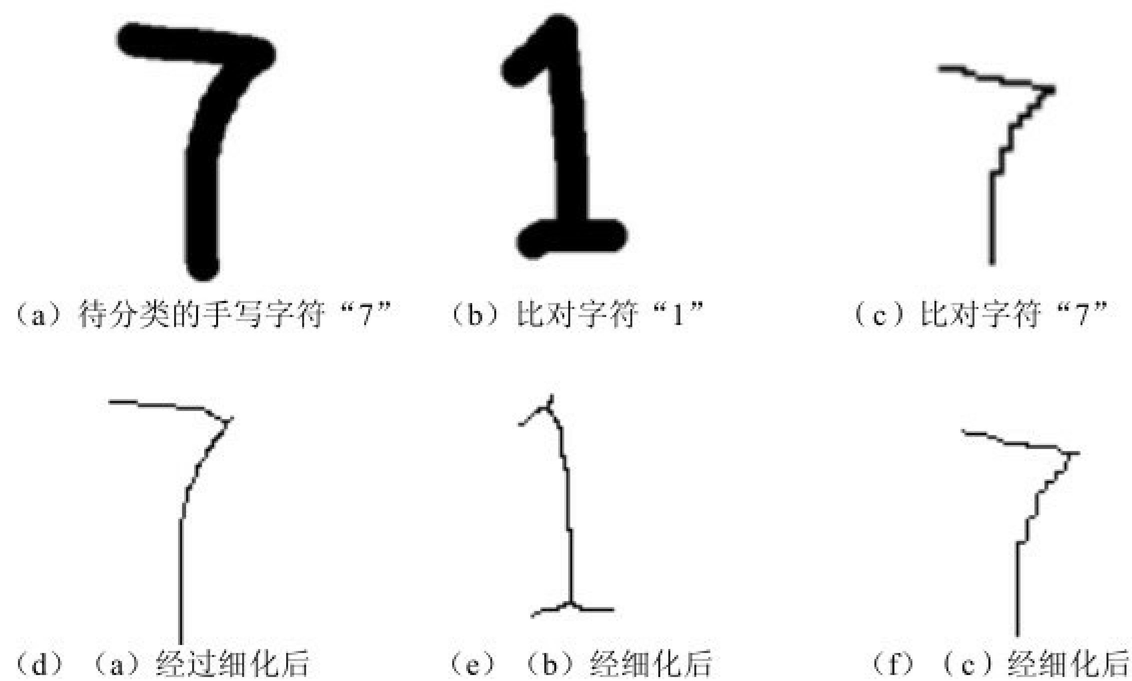
▲图10.25 统计连通区个数

10.3.5 细化算法

“骨架”是指一幅图像的骨骼部分，它描述物体的几何形状和拓扑结构，是重要的图像描绘子之一。计算骨架的过程一般称为“细化”或“骨架化”，在包括文字识别、工业零件形状识别以及印刷电路板自动检测在内的很多应用中，细化过程都发挥着关键作用。通常，对我们感兴趣的目标物体进行细化有助于突出目标的形状特点和拓扑结构，并且减少冗余的信息量。

[例10.7]手写字符的细化。

同类物体的由于其线条粗细不同而显得差别很大（图10.26（a）和（c）中的“7”），这无疑会给后续的识别任务带来困扰，例如对于10.26中的图像来说识别程序很可能会认为（a）中的“7”更像图10.26（b）中的“1”而不是图10.26（c）中的“7”。但将它们的形状细化之后，归一化为相同的宽度，如1个像素宽，此时我们发现图10.26（d）～（f）的3幅数字骨架图像所体现出来的完全是数字本身的几何形状，在这些细化后的图像中选择适当的特征进行分类将得到理想的结果。



（a）待分类的手写字符“7”　（b）比对字符“1”　（c）比对字符“7”

（d）（a）经过细化后　（e）（b）经细化后　（f）（c）经细化后

▲图10.26 手写数字字符的细化

下面我们来看一下细化算法的实现思路。

考虑某图像中的一个3×3的区域，对其中各点标记名称P1,P2,…,P8，如图10.27（a）所示。这里规定以1表示黑色，0表示白色，则如果中心P1=1（黑点），下面4个条件同时满足，就删除P1(令P1=0)。

（1）2-NZ(P1)-6。

（2）Z0(P1)=1。

（3）P2*P4*P8=0或者Z0(P1)≠1。

（4）P2*P4*P6=0或者Z0(P4)≠1。

其中的标记NZ（P）表示P点的8邻域中1的数目；而Z0（P）可按照如下方式计算。

$P_{-1,-1}$	$P_{-1,0}$	$P_{-1,1}$
$P_{0,-1}$	P	$P_{0,1}$
$P_{1,-1}$	$P_{1,0}$	$P_{1,1}$

（1）令 nCount = 0

（2）如果$p_{-1,0}$=0并且$p_{-1,-1}$=1

nCount ++

（3）如果$p_{-1,-1}$=0并且$p_{0,-1}$=0

nCount ++

（4）如果$p_{0,-1}$=0并且$p_{1,-1}$=1

nCount ++

（5）如果$p_{1,-1}$=0并且$p_{1,0}$=1

nCount ++

（6）如果$p_{1,0}$=0 并且$p_{1,1}$=1

nCount ++

（7）如果$p_{1,1}$=0 并且$p_{0,1}$=1

nCount ++

（8）如果$p_{0,1}$=0并且$p_{-1,1}$=1

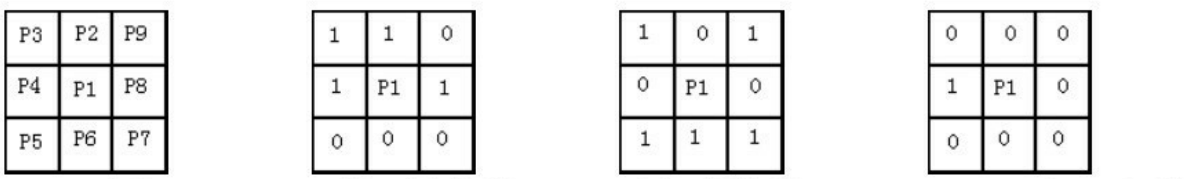
nCount ++

（9）如果$p_{-1,1}$=0并且$p_{-1,0}$=1

nCount ++

（10）Z0(P)=nCount

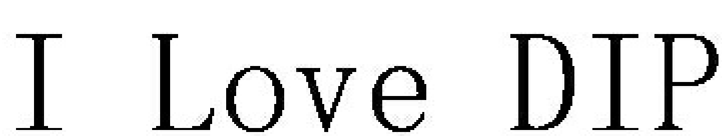
对图像中的每一个点重复这一步骤，直到所有的点都不可删除为止。图10.25给出了细化算法的示意。



（a）3×3 邻域　（b）删除 P1 会分割区域　（c）删除 P1 会缩短边缘　（d）2≦NZ(P1)≦6，但 P1 不可删除

▲图10.27 细化算法示意

在Matlab中，可以使用bwmorph 函数方便地实现图像细化。关于bwmorph 函数将在 10.3.8小节中介绍，一个图像细化的具体效果如图10.28所示。



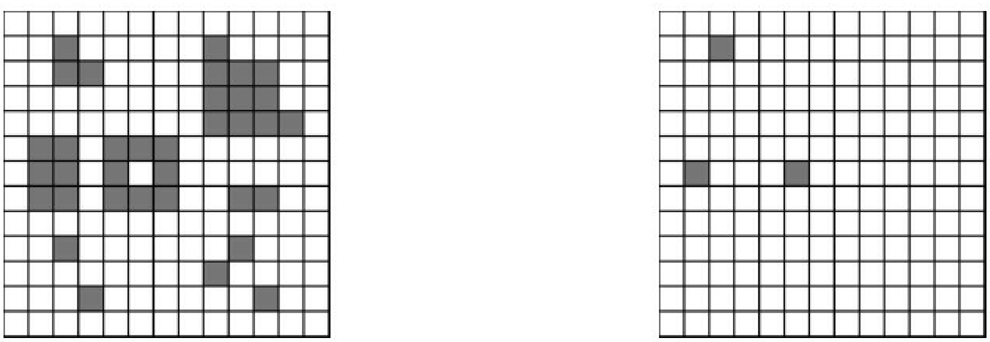
▲图10.28 图像细化

10.3.6 像素化算法

细化适用于和物体拓扑结构或形状有关的应用，如前述的手写字符识别。但有时我们关心的是目标对象是否存在、他们的位置关系、或者是个数，这时在预处理中加入像素化步骤就会给后续的图像分析带来极大的方便。

像素化操作首先找到二值图像中所有的连通区域，然后用这些区域的质心作为这些连通区域的代表，即将1个连通区域像素化为位于区域质心位置的1个像素。

有时还可以进一步引入一个低阈值 lowerThres 和一个高阈值 upperThres 用来指出图像中我们感兴趣的对象连通数（连通分量中的像素数目）的大致范围，从而只像素化图像中大小介于lowerThres 和 upperThres之间的连通区域，而连通数低于lowerThres或高于upperThres的对象都将被滤除，这就相当于使算法同时具有了过滤噪声的能力，如图10.29所示。



（a）原始图像 A

（b）A 经像素化处理之后，注意 A 中<lowerThres 和>upperThres 的连通区域均被滤出了，在这之间的区域都像素化到了区域的质心

▲图10.29 像素化效果模拟，lowerThres = 3，upperThres = 10

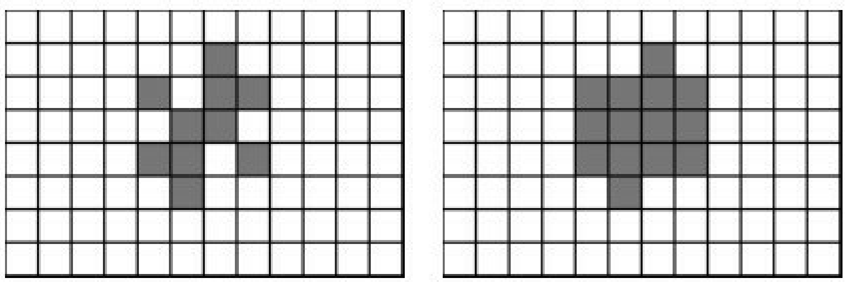
实现细化算法的核心是找到连通区域，接下来只需要根据 lowerThres和upperThres参数把内涵像素数不满足要求的连通区域滤除，并计算保留的连通区域的几何中心。由于连通区域的Matlab实现已在10.3.4小节给出，上述实现过程就留给读者作为练习。

10.3.7 凸壳

理论基础

如果连接物体A内任意两点的直线段都在A的内部，则称A是凸的。任意物体A的凸壳H是包含A的最小凸物体。

我们总是希望像素化算法能够找到物体的质心来代表该物体，但在实际中，可能由于光照不均等原因导致图像在二值化后，物体本身形状发生缺损，像素化算法就无法找到物体真正的质心。此时可适当地进行凸壳处理，弥补凹损，算法会找到包含原始形状的最小凸多边形，如图10.30所示。



▲图10.30 凸壳运算效果模拟

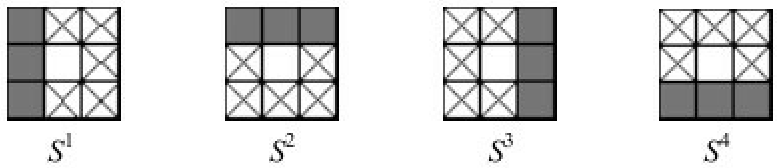
令s^i，i=1,2,3,4,表示图10.31中的4个结构元素。则凸壳的计算过程如下。

$$X_k^i=(X_{k-1}^i \circledast S^i)\cup A \qquad i=1,2,3,4 \qquad k=1,2,3,\ldots$$

X_0^i =A。现在令D^i =x_{conv}，这里下标“conv”表示在$X_k^i=X_{k-1}^i$时收敛。A的凸壳为

$$C(A)=\bigcup_{i=1}^{4} D^i$$

用结构元素s^1对A反复进行击中或击不中变换，直到不再发生进一步变化时，与A求并集，结果记作D^1。由结构元素s^i（i=2,3,4）和A进行相同的运算可得D^i（i=2,3,4）。最后，4个D的并组成了A的凸壳。



▲图10.31 凸壳计算中的4个结构元素（图中“X”表示不关心该像素的值）

为确保在上述的生长过程中凸壳不会超出凸性所需的最小尺寸很多，可以限制其生长，以便凸壳不会超出初始时包含物体A的最小矩形。

凸壳的处理效果如图10.32所示。



▲图10.32 凸壳计算结果

10.3.8 bwmorph函数

本章的很多形态学操作都可由IPT函数bwmorph实现，该函数的调用语法为

Iout = bwmorph(I, operation, n);

参数说明

- I为输入二值图像。
- operation是一个指定操作类型的字符串，常用的合法取值如表

10.2所示。

● 可选参数n是一个正整数，用于指定将被重复的操作次数，默认为1。当n=Inf时表示重复操作一直到图像停止改变为止。

返回值

● Iout为经过n次由operation参数指定的形态学操作后的输出图像。

表10.2 operation参数常用取值及意义

合法取值	功能描述
'bridge'	桥接由单个像素缝隙分隔的前景像素
'clean'	清除孤立的前景像素
'diag'	围绕对角线相连的前景像素进行填充
'fill'	填充单个像素的孔洞
'hbreak'	去掉前景中的H形连结
'majority'	如点 *P* 的8邻域中一半以上的像素为前景像素，则使 *P* 也为前景像素；否则使 *P* 为背景像素
'remove'	去处内部像素（无背景像素相邻的前景像素）
'shrink'	将物体收缩为一个点或者带洞的环形
'skel'	骨骼化图像
'spur'	去除“毛刺”
'thicken'	粗化物体
'thin'	将物体细化至最低限度相连的线形

10.4 灰度图像中的基本形态学运算

本节我们把二值图像的形态学处理扩展到灰度图像的基本操作，包括灰度膨胀、灰度腐蚀、灰度开和灰度闭。此外，10.4.4 小节还将介绍一个灰度形态学的经典应用——顶帽变换（top-hat），用以解决图像的光照不均问题。

10.4.1 灰度膨胀及其实现

1．理论基础

令F表示灰度图像，S为结构元素，使用S对F进行膨胀，记作F⊕S，形式化地定义为

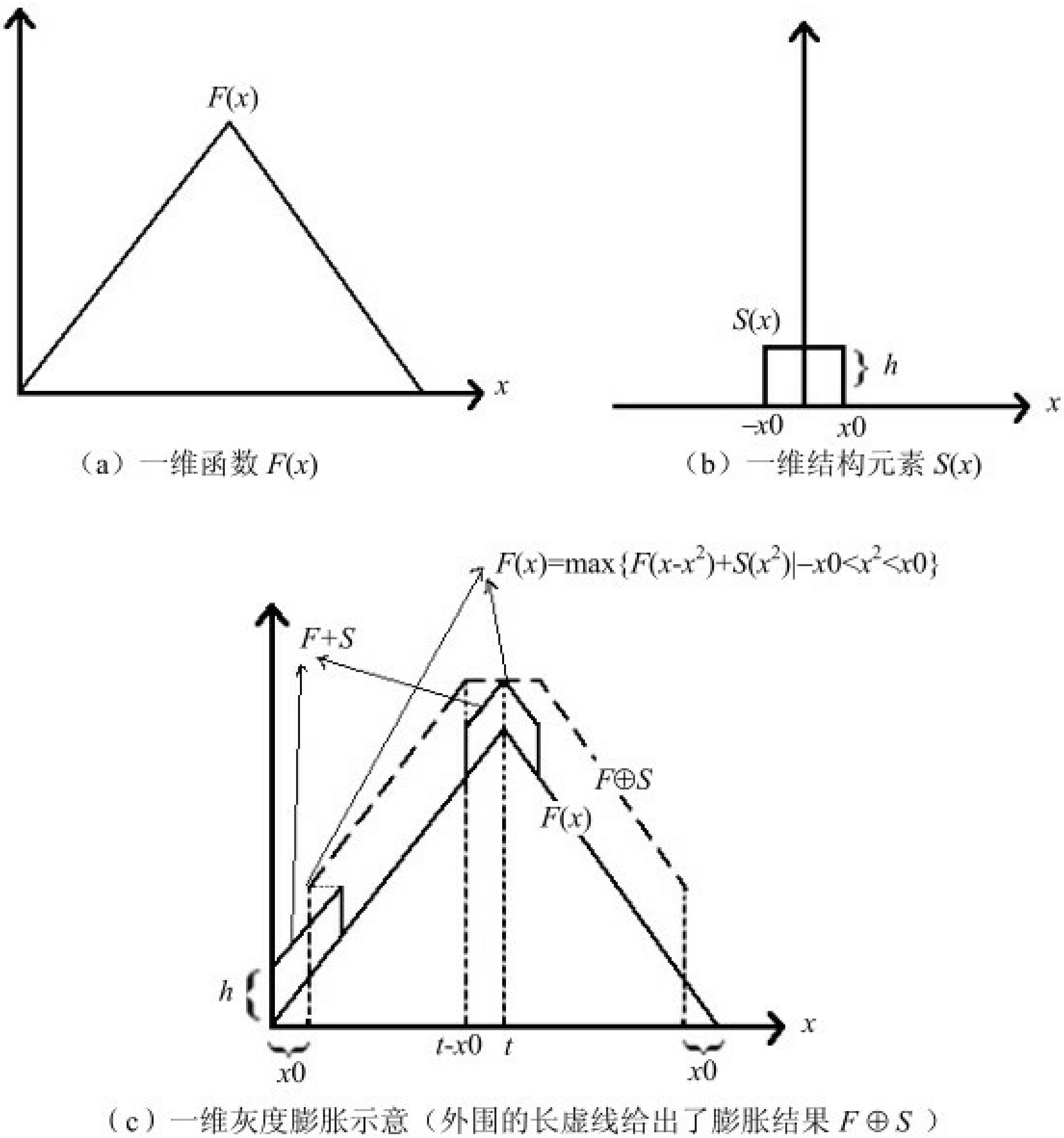
$$(F \oplus S)(x,y) = \max\{F(x-x',y-y') + S(x',y') \mid (x',y') \in D_S\} \qquad (10\text{-}9)$$

式中：D_S是S的定义域。

计算过程相当于让结构元素S关于原点的镜像$\hat{S}$在图像F的所有位置上滑过，而在此过程中要保证(x+x’, y+y’)始终灰度图像F之内。膨胀结果F⊕S在其定义域内每一点（x, y）处的取值为以（x，y）为中心，在$\hat{S}$规定的局部邻域内F与$\hat{S}$之和的最大值。例如，对于正方形结构元素S（−x0～x0,−y0～y0，中心为(0,0))，膨胀结果为F与$\hat{S}$之和在局部邻域[x−x0,x+x0;y−y0,y+y0]内的最大值。注意这一过程与卷积有许多相似之处，只是用最大值运算代替卷积求和，用加法用算代替卷积乘积。

与二值形态学不同的是，F(x,y)和S(x,y)不再是只代表形状的集合，而是二维函数，它们的定义域指明了其形状，此外它们还有高度信息，由函数值指出。

图 10.33 给出了一维灰度膨胀的示意图，如 10.33（c）中，在 x0 处F⊕S (x0)的值即为max{F(x0− x ')+S(x ') |−x0< x '< x0}，具体到这里应为h+F(x0+x0)；而在t处 F⊕S (t) =max{F(t− x ')+S(x ') |−x0< x '< x0} =F(t)+h；注意造成F⊕S (x)中间部分呈水平走势的原因是F(x)在x=t处达到了局部最大值。



（a）一维函数 $F(x)$

（b）一维结构元素 $S(x)$

（c）一维灰度膨胀示意（外围的长虚线给出了膨胀结果 $F\oplus S$ ）

▲图10.33 灰度膨胀示意图

除了具有高度的结构元素外，实际应用中使用更多的是一种平坦(高度为0)的结构元素，这种结构元素只能由0和1组成，为1的区域指明了运算涉及的范围。实际上，二值形态学中的结构元素可视为一种特殊（高度为0）的灰度形态结构元素。当应用这种结构元素时，灰度膨胀完全变成了局部最大值运算，其计算公式可简化为

$$F \oplus S(x,y) = \max\{f(x-x', y-y') \mid ((x', y') \in D_S\} \qquad (10\text{-}10)$$

2．Matlab实现

只要以灰度图像和相应的灰度膨胀结构元素为参数调用imdilate函数即可实现灰度膨胀。平坦结构元素的创建方法与二值形态学中相同；而非平坦结构元素也可通过strel函数以如下方式创建。

SE = strel(NHOOD, HEIGHT); %创建非平坦的结构元素

参数说明

● NHOOD为指明结构元素定义域的矩阵，只能由0和1组成。

● HEIGHT是一个与NHOOD具有相同尺寸的矩阵，指出了对应于NHOOD中每个元素的高度。

返回值

● SE为返回的非平坦结构元素。

下面结合例10.8来说明strel和imdilate函数在灰度膨胀中的用法。

[例10.8]灰度膨胀。

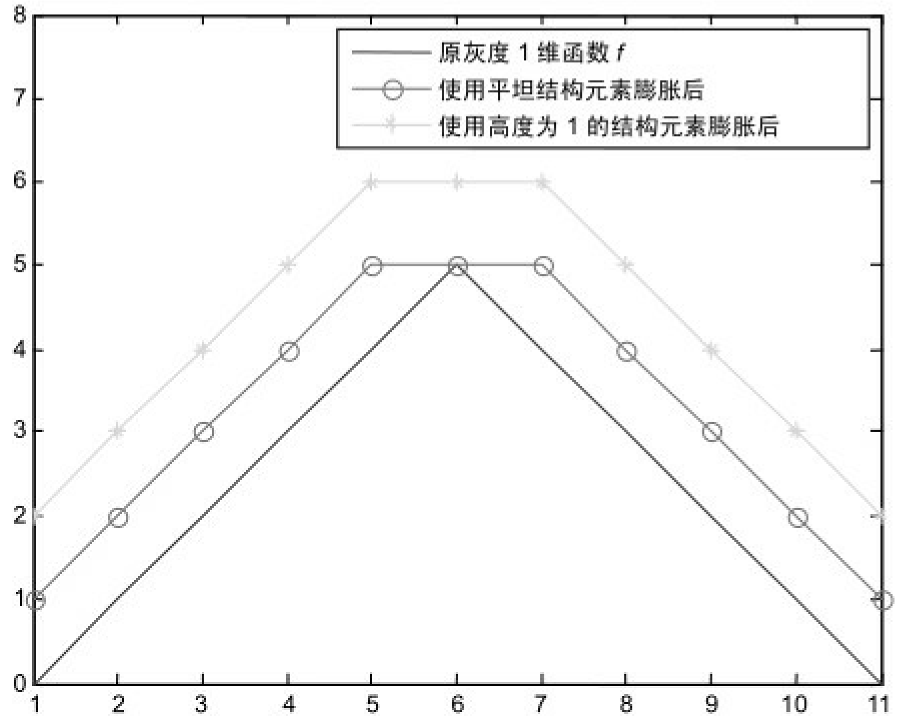
分别用高度为1和平坦的结构元素实现灰度膨胀，并绘制出膨胀前后的函数图形。

相应的Matlab实现代码如下。

```
>> f = [0 1 2 3 4 5 4 3 2 1 0];
>> figure, h_f = plot(f);
>>
>> seFlat = strel([1 1 1]) % 构造平坦(高度为0)的结构元素
seFlat =
Flat STREL object containing 3 neighbors.
Neighborhood:
   1    1    1
>> fd1 = imdilate(f, seFlat); % 使用平坦的结构元素灰度膨胀
```

```
>> hold on, h_fd1 = plot(fd1, '-ro');
>> axis([1 11 0 8])
>>
>> seHeight = strel([1 1 1], [1 1 1]) % 注意此处strel的用法，第一个参数的元素为0或1，表示结
构元素的区域范围（形状），第二个参数表示结构元素中各个元素的高度
seHeight =
Nonflat STREL object containing 3 neighbors.
Neighborhood:
   1   1   1
Height:
   1   1   1
>> fd2 = imdilate(f, seHeight); %使用具有高度的结构元素的灰度膨胀
>> hold on, h_fd2 = plot(fd2, '-g*');
>> legend('原灰度1维函数f', '使用平坦结构元素膨胀后', '使用高度为1的结构元素膨胀后');
```

上述程序的运行结果如图10.34所示。



▲图10.34 灰度膨胀

10.4.2 灰度腐蚀及其实现

理论基础

令F表示灰度图像，S为结构元素，使用S对F进行腐蚀，记作F $\ominus$ S，形式化地定义为

$$(F \ominus S)(x,\ y)\ =\ \min\{F(x+x', y+y') - S(x', y') \mid (x', y') \in D_S\} \qquad (10\text{-}11)$$

其中，D_S是S的定义域，计算过程相当于让结构元素在图像F的所有位置上滑过，而在此过程中要保证(x+x’，y+y’)始终在图像F之内。腐蚀结果F $\ominus$ S在其定义域内每一点（x,y）处的取值为以（x，y）为中心，在S规定的局部邻域内F与S之差的最小值。

与二值形态学不同的是，F(x, y)和S(x, y)不再是只代表形状的集合，而是二维函数，它们的定义域指明了其形状，它们的值指出了高度信息。

[例10.9]灰度腐蚀。

分别用高度为1和平坦的结构元素实现灰度腐蚀，并绘制出腐蚀前后的函数图形。

相应的Matlab代码如下。

```
f = [0 1 2 3 4 5 4 3 2 1 0];
>> figure, h_f = plot(f);
>> seFlat = strel([1 1 1]) % 构造平坦(高度为0)的结构元素
seFlat =
Flat STREL object containing 3 neighbors.
Neighborhood:
    1    1    1
>> fe1 = imerode(f, seFlat); %使用平坦结构元素的灰度腐蚀
>> hold on, h_fe1 = plot(fe1, '-ro');
>> axis([1 11 0 8])
>>
>> seHeight = strel([1 1 1], [1 1 1]) % 注意此处strel的用法，第一个参数的元素为0或1，表示结
构元素的区域范围（形状），第二个参数表示结构元素的高度
seHeight =
Nonflat STREL object containing 3 neighbors.
Neighborhood:
    1    1    1
Height:
    1    1    1
>> fe2 = imerode(f, seHeight); %使用具有高度的结构元素的灰度腐蚀
>> hold on, h_fe2 = plot(fe2, '-g*');
```

>> legend('原灰度1维函数f', '使用平坦结构元素腐蚀后', '使用高度为1的结构元素腐蚀后');

上述程序的运行结果如图10.35所示。

[例10.10]灰度膨胀和灰度腐蚀的效果比较。

采用单位高度的3×3结构元素对图像lena.bmp分别进行灰度膨胀和灰度腐蚀的Matlab代码如下。

>> I = imread('../lena.bmp');

>> seHeight = strel(ones(3, 3), ones(3, 3)) % 3*3正方形单位高度的结构元素

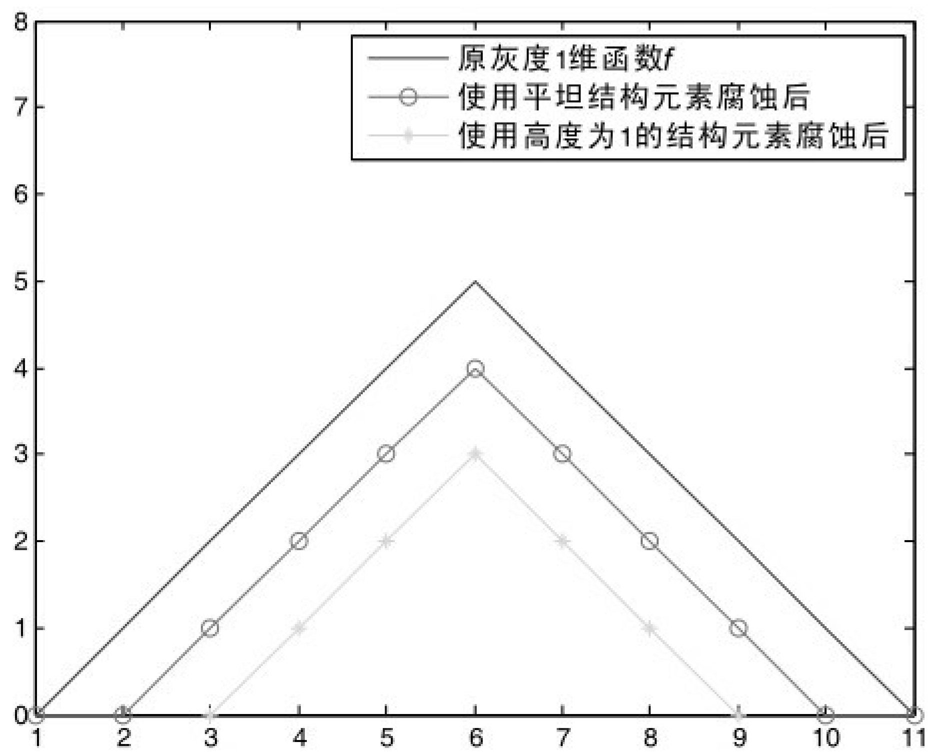
seHeight =

Nonflat STREL object containing 9 neighbors.

Neighborhood:

1 1 1

1 1 1



▲图10.35 灰度腐蚀

```
   1    1    1
Height:
   1    1    1
   1    1    1
   1 1    1
>> Idil = imdilate(I, seHeight);
>> Iero = imerode(I, seHeight);
>> subplot(1, 3, 1), imshow(I) %得到图10.36（a）
subplot(1, 3, 2), imshow(Idil) %得到图10.36（b）
subplot(1, 3, 3), imshow(Iero) %得到图10.36（c）
```

上述程序的运行结果如图10.36所示。



（a）原图 lena.bmp　　（b）图（a）经过灰度膨胀　　（c）图（a）经过灰度腐蚀

▲图10.36 灰度膨胀与灰度腐蚀

我们看到在结构元素的值均大于零的情况下，灰度膨胀的输出图像总体上比输入图像更亮，这是局部最大值运算作用的结果，此外原图像中一些能够包含于结构元素的暗细节（如一部分帽子的褶皱和尾穗）被完全消除，其余的大部分暗部细节也都得到了一定程度上的减少。而灰度腐蚀的作用正好相反，输出图像比输入图像更暗，如果输入图像中的

亮部细节比结构元素小，则亮度会得到削弱。

10.4.3 灰度开、闭运算及其实现

1．理论基础

类似于二值形态学，可以在灰度腐蚀和膨胀的基础上定义灰度开和闭运算。灰度开运算就是先灰度腐蚀后灰度膨胀，而灰度闭运算是先灰度膨胀后灰度腐蚀，下面分别给出定义。

● 灰度开运算

使用结构元素s对图像f灰度进行开运算，记作f ◦ s，可表示为

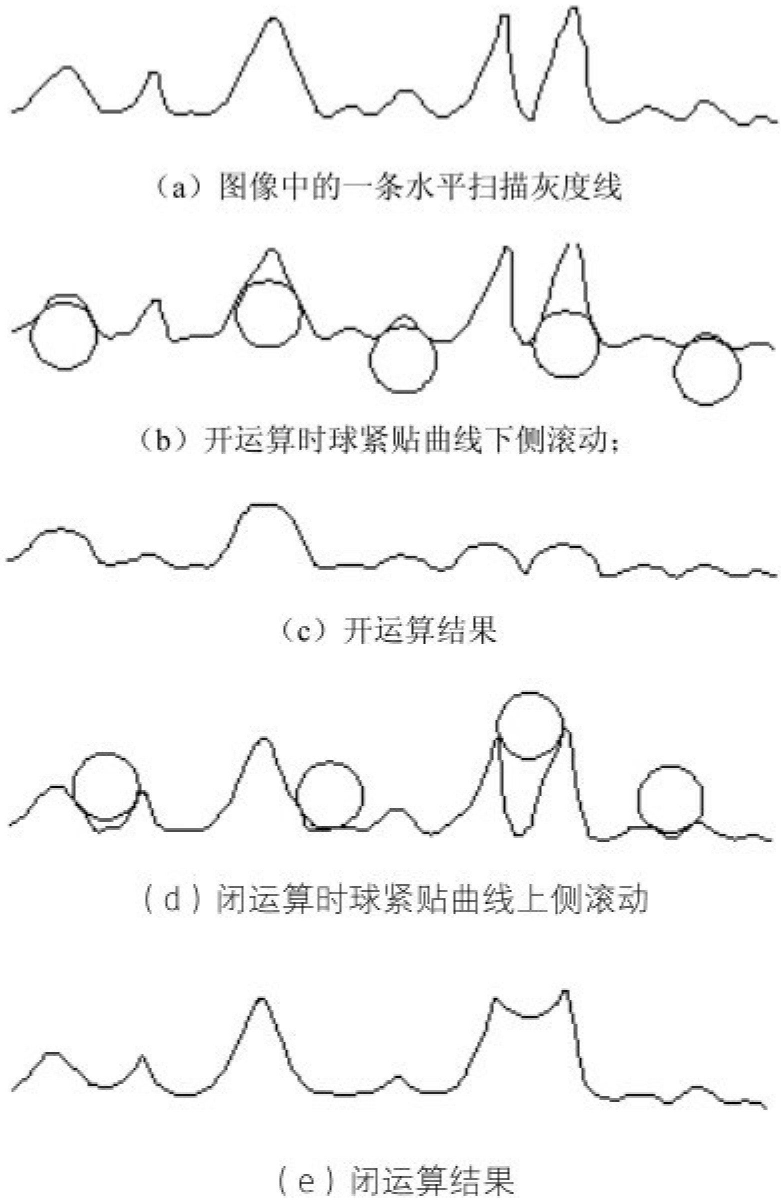
$$f \circ s = (f \ominus s) \oplus s \tag{10-12}$$

● 灰度闭运算

使用结构元素s对图像f进行灰度闭运算，记作f•s，可表示为

$$f \bullet s = (f \oplus s) \ominus s \tag{10-13}$$

假设我们有一个球形的结构元素s，开运算相当于推动球沿着曲面的下侧面滚动，使得球体可以紧贴着下侧面来回移动，直到移动位置覆盖了整个下侧面。此时球体的任何部分能够达到的最高点构成了开运算f◦s的曲面；而闭运算则相当于让球体紧贴在曲面的上表面滚动，此时球体任何部分所能达到的最低点即构成了闭运算f •s的曲面。图10.37形象地说明了这一过程，图10.37（a）所示为图像中的一条水平像素线；图10.37（b）和图10.37（d）分别给出了球紧贴着该像素线的下侧和上侧滚动的情况；而图10.37（c）和图10.37（e）则展示了滚动过程中球的最高点形成的曲线，它们分别是开、闭运算的结果。



▲图10.37 灰度开、闭运算示意图

2．Matlab实现

使用imopen和imdilate同样可以对灰度图像进行开、闭运算，用法与灰度腐蚀和膨胀类似，这里就不再专门介绍了。

在实际应用中，开运算常常用于去除那些相对于结构元素s而言较小的高灰度区域（球体滚不上去），而对于较大的亮区域影响不大（球体可以滚上去）。虽然首先进行的灰度腐蚀会在去除图像细节的同时使

得整体灰度下降，但随后的灰度膨胀又会增强图像的整体亮度，因此图像的整体灰度基本保持不变；而闭运算常用于去除图像中的暗细节部分，而相对地保留高灰度部分不受影响，如图10.38所示。


（a）开运算处理图像


（b）闭运算处理图像

▲图10.38 灰度开、闭运算

10.4.4 顶帽变换（top-hat）及其实现

理论基础

作为灰度形态学的重要应用之一，这里学习一种非均匀光照问题的解决方案——顶帽变换技术（top-hat）。图像f的顶帽变换h定义为图像f与图像f的开运算之差，可表示为

$$h = f - (f \circ s) \qquad (10\text{-}14)$$

[例10.11]灰度形态学综合应用——顶帽变换。

图10.39（a）中是Matlab中自带的米粒图像rice.png，相对高灰度的米粒分散于整体的暗背景中，注意到图像上中部分的背景明显要比下部的背景区域亮一些，这正是由成像时不均匀的光照引起的。我们希望能

够通过阈值化把物体（米粒）与背景分离开来，从而进一步研究物体的性质，如个数和位置关系等。

然而，在图像整体灰度不均的情况下，直接对图10.39（a）进行的阈值化将难以得到满意的效果，图10.39（b）中给出了用自适应最优阈值法进行分割的结果，可以看到底部的一些米粒分割效果较差。为了理解起来更加直观，在图10.39（c）中给出了图像f的三维可视化效果，f中的米粒对应于图 10.39（c）图中一个个“山峰”，注意到这些“峰”位于一个“斜坡”上，换言之，它们的“地基”并不处于同一高度，这正是由f中不均匀的背景造成的。

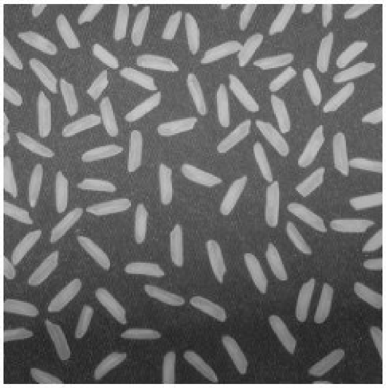
通过运用合适的结构元素s与f进行开运算可以消除这些峰值。f中的米粒都具有近似大小，因此只要选择直径比米粒的短轴稍大的圆形结构元素就可以达到目的。图10.39（d）给出了采用半径为15的圆形结构元素与f进行开运算后得到图像的三维可视化效果。想象你水平地托着一个圆盘从图10.39（c）中曲面的下侧走过，这个过程中始终让圆盘顶着曲面的下侧，最终圆盘最高点形成的曲面就是图10.39（d）中的曲面。由于圆盘比米粒凸起的“山峰”略大，这实际上得到了一个消除了峰值的背景曲面。显然，从原图像f中减去这个不均匀的背景就得到了亮度比较均匀的米粒图像图10.39（e），其三维可视化效果如图10.39（f）所示，这些“山峰”已经基本处于同一高度。图10.39（g）给出了图10.39（e）经过灰度拉伸的效果；图10.39（h）是对图10.39（g）中的图像进行阈值分割的结果，已经非常理想了。

相应的Matlab代码如下。

```
>> I = imread('rice.png');
>> subplot(2, 4, 1), imshow(I, []);%得到图10.39（a）
>> thresh = graythresh(I) %自适应确定阈值
thresh =
  0.5137
```

```
>> Ibw = im2bw(I, thresh);
>> subplot(2, 4, 2), imshow(Ibw, []);%得到图10.39（b）
>> subplot(2, 4, 3), surf(double(I(1:8:end,1:8:end))),zlim([0 255]),colormap gray;%显示I的3维可视化效果，（c）图
>>
>> bg = imopen(I,strel('disk',15));%半径为15的圆形结构元素进行灰度开运算提取背景曲面
>> subplot(2, 4, 4), surf(double(bg(1:8:end,1:8:end))),zlim([0 255]), colormap gray;
%显示背景曲面的三维可视化效果，图10.39（d）
>>
>> Itophat = imsubtract(I, bg); %顶帽变换
>> subplot(2, 4, 5), imshow(Itophat); %得到图10.39（e）
>> subplot(2, 4, 6), surf(double(Itophat(1:8:end,1:8:end))),zlim([0 255]); %显示顶帽变换图像的三维可视化效果
>> I2 = imadjust(Itophat);%对比度拉伸
>> subplot(2, 4, 7), imshow(I2); %得到图10.39（f）
>>
>> thresh2 = graythresh(I2) %自适应确定阈值
thresh2 =
  0.4843
>> Ibw2 = im2bw(I2, thresh2); %得到图10.39（g）
>> subplot(2, 4, 8), imshow(Ibw2); %得到图10.39（h）
```

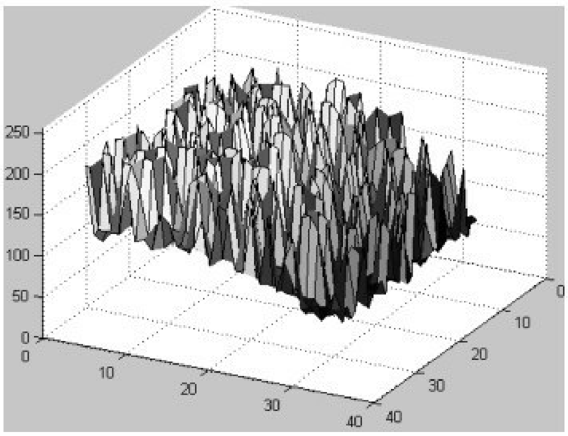
上述程序的运行结果如图10.39所示。



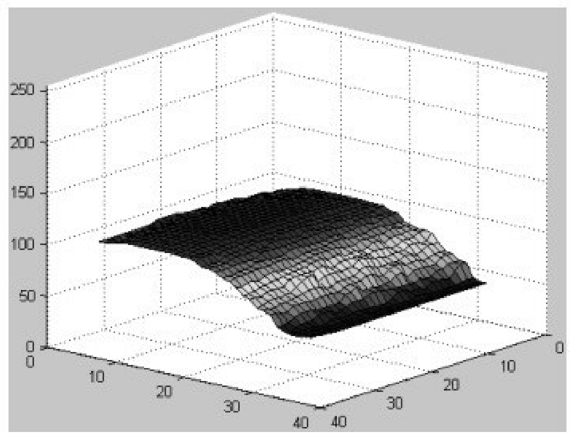
（a）原图像 f



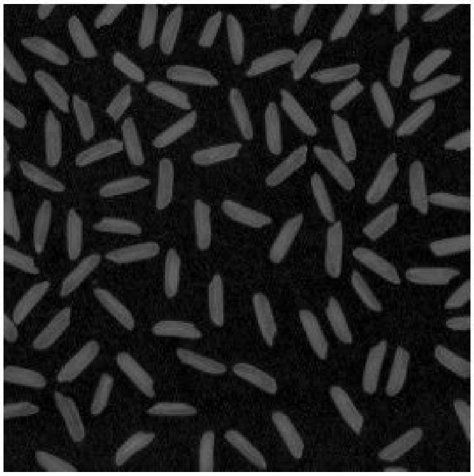
（b）对（a）图像 f 经过自适应阈值处理



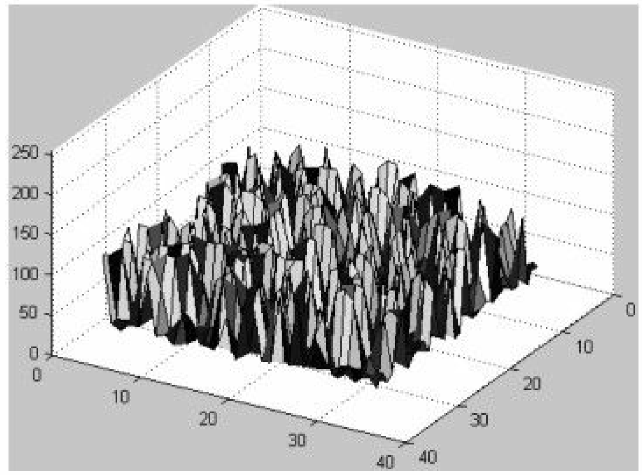
（c）图像 f 的三维可视化效果



（d）对图像 f 进行灰度开运算得到的背景区面



（e）顶帽变换图像 $f-(f\circ s)$



（f）顶帽变换图像 $f-(f\circ s)$ 的三维可视化效果



（g）将图像 $f-(f\circ s)$ 对比度拉伸



（h）对顶帽变换后图像进行二值化

▲图10.39 顶帽变换处理光照不均的图像

10.5 小结

在本章中，我们介绍了形态学的基本概念并学习了多种常见的形态学算法及其典型应用。合理运用这些技术能够帮助我们从图像中提取感兴趣的特征，这常常和下一章中将要介绍的图像分割技术结合在一起，为最终得到能够直接用于图像识别的数值或向量特征铺平了道路。事实上，在下一章中还会发现一些基于形态学的分割方法，如著名的分水岭

算法。

第11章 图像分割

图像分割是指将图像中具有特殊意义的不同区域划分开来，这些区域是互不相交的，每个区域满足灰度、纹理、彩色等特征的某种相似性准则。图像分割是图像的分析过程中最重要的步骤之一，分割出的区域可以作为后续特征提取的目标对象。

本章的知识和技术热点

- 基于梯度的Sobel、Prewitt和Roberts算子的边缘检测
- LoG边缘检测算法
- Canny边缘检测算法
- Hough变换和直线检测
- 阈值分割技术
- 基于区域的图像分割技术

本章的典型案例

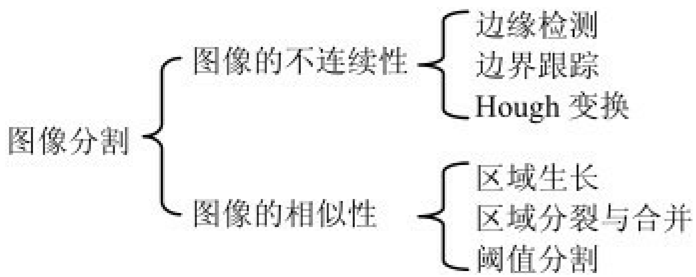
- 基于LoG和Canny算子的精确边缘检测
- 基于Hough变换的直线检测
- 图像的四叉树分解
- 基于形态学分水岭算法的图像分割

11.1 图像分割概述

图像分割的方法和种类有很多，有些分割算法可以直接运用于大多数图像，而另一些则只适用于特殊类别的图像，要视具体情况来决定。一般采用的方法有边缘检测（edge detection）、边界跟踪（edge tracing）、区域生长（region growing）、区域分离和聚合等。

图像分割算法一般基于图像灰度值的不连续性或其相似性。不连续

性是基于图像灰度的不连续变化分割图像，例如图像的边缘，有边缘检测、边界跟踪等算法；相似性是依据事先制定的准则将图像分割为相似的区域，如阈值分割、区域生长等，如图11.1所示。



▲图11.1 图像分割的分类

图像分割在实际的科学研究和工程技术领域有着广泛的应用。在工业上，应用于矿藏分析、无接触式检测、产品的精度和纯度分析等；生物医学上，应用于计算机断层图像CT、X光透视、核磁共振、病毒细胞的自动检测和识别等；交通上，应用于车辆检测、车种识别、车辆跟踪等；另外，在机器人视觉、神经网络、身份鉴定、图像传输等各个领域都有着广泛的应用。

11.2 边缘检测

图像的边缘是图像的最基本特征，边缘点是指图像中周围像素灰度有阶跃变化或屋顶变化的那些像素点，即灰度值导数较大或极大的地方。图像属性中的显著变化通常反映了属性的重要意义和特征。

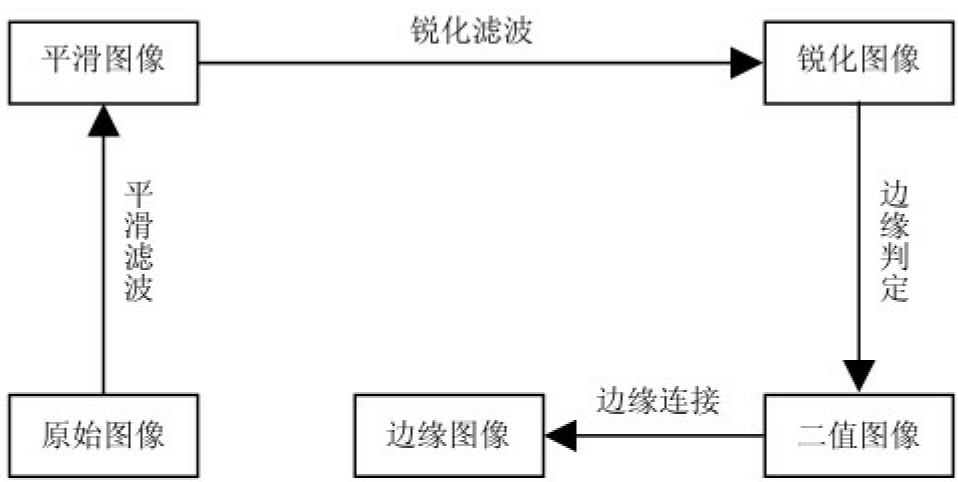
边缘检测是图像处理和计算机视觉中的基本问题，边缘检测的目的是标识数字图像中亮度变化明显的点。我们曾在5.5节中讨论了一些可以用于增强边缘的图像锐化方法，本节介绍如何将它们用于边缘检测；此外，还将学习一种专门用于边缘检测的Canny算子。

11.2.1 边缘检测概述

边缘检测可以大幅度地减少数据量，并且剔除了那些被认为不相关的信息，保留了图像重要的结构属性。

1．边缘检测的基本步骤

边缘检测的步骤如图11.2所示。



▲图11.2 边缘检测

（1）平滑滤波：由于梯度计算易受噪声影响，因此第一步是用滤波去除噪声。但是，降低噪声的平滑能力越强，边界强度的损失越大。

（2）锐化滤波：为了检测边界，必须确定某点邻域中灰度的变化。锐化操作加强了存在有意义的灰度局部变化位置的像素点。

（3）边缘判定：在图像中存在许多梯度不为零的点，但是对于特定应用，不是所有点都有意义。这就要求我们根据具体情况选择和去除处理点，具体的方法包括二值化处理和过零检测等。

（4）边缘连接：将间断的边缘连接成为有意义的完整边缘，同时去处假边缘。主要方法是Hough变换。

2．边缘检测方法的分类

通常可将边缘检测的算法分为2类：基于查找的算法和基于零穿越

的算法。除此之外，还有Canny边缘检测算法、统计判别方法等。

● 基于查找的方法通过寻找图像一阶导数中的最大和最小值来检测边界，通常是将边界定位在梯度最大的方向，是基于一阶导数的边缘检测算法。

● 基于零穿越的方法通过寻找图像二阶导数零穿越来寻找边界，通常是拉普拉斯过零点或者非线性差分表示的过零点，是基于二阶导数的边缘检测算法。

基于一阶导数的边缘检测算子包括Roberts算子、Sobel算子、Prewitt算子等，它们都是梯度算子；基于二阶导数的边缘检测算子主要是高斯-拉普拉斯边缘检测算子。我们将在11.2.2小节对它们进行中介绍。

11.2.2 常用的边缘检测算子

1．梯度算子

几个最常用梯度算子的模板如图11.3所示。

$$\begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \qquad \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$$

（a）Roberts 模板

$$\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \qquad \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

（b）水平（左）和竖直（右）的 Sobel 模板

$$\begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix} \qquad \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$$

（c）水平（左）和竖直（右）的 Prewitt 模板

▲图11.3 梯度算子模板

Roberts 算子利用局部差分算子寻找边缘，边缘定位精度较高，但容易丢失一部分边缘，同时由于图像没经过平滑处理，因此不具备抑制噪声的能力。该算子对具有陡峭边缘且含噪声少的图像效果较好。

Sobel算子和Prewitt算子都考虑了邻域信息，相当于对图像先做加权平滑处理，然后再做微分运算，所不同的是平滑部分的权值有些差异，因此对噪声具有一定的抑制能力，但不能完全排除检测结果中出现的虚假边缘。虽然这两个算子边缘定位效果不错，但检测出的边缘容易出现多像素宽度。

2．高斯-拉普拉斯算子

在5.5.3小节已经介绍了拉普拉斯算子，但是由于它是一个二阶导数，对噪声具有无法接受的敏感性，而且其幅值会产生双边缘，另外，边缘方向的不可检测也是拉普拉斯算子的缺点之一，因此，一般不以其原始形式作用于边缘检测。

为了弥补拉普拉斯算子与生俱来的缺陷，美国学者Marr提出了一种算法，在运用拉普拉斯算子之前一般先进行高斯低通滤波，可表示为

$$\nabla^2[G(x,y)*f(x,y)] \tag{11-1}$$

其中f(x, y)为图像，G(x, y)为高斯函数，表示为

$$G(x,y)=\frac{1}{2\pi\sigma^2}\exp(-\frac{x^2+y^2}{2\sigma^2}) \tag{11-2}$$

其中σ是标准差。用高斯函数卷积模糊一幅图像，图像模糊的程度是由σ决定的。

由于在线性系统中卷积与微分的次序可以交换，由式（11-1）得

$$\nabla^2[G(x,y)*f(x,y)]=\nabla^2 G(x,y)*f(x,y) \tag{11-3}$$

式（11-3）说明了可以先对高斯算子进行微分运算，然后再与图像f(x, y)卷积，其效果等价于在运用拉普拉斯算子之前首先进行高斯低通滤波。

计算式（11-2）的二阶偏导为

$$\frac{\partial^2 G(x,y)}{\partial x^2}=\frac{1}{2\pi\sigma^4}\left[\frac{x^2}{\sigma^2}-1\right]\exp(-\frac{x^2+y^2}{2\sigma^2}) \tag{11-4}$$

$$\frac{\partial^2 G(x,y)}{\partial y^2}=\frac{1}{2\pi\sigma^4}\left[\frac{y^2}{\sigma^2}-1\right]\exp(-\frac{x^2+y^2}{2\sigma^2}) \tag{11-5}$$

可得

$$\nabla^2 G(x,y)=-\frac{1}{\pi\sigma^4}\left[1-\frac{x^2+y^2}{2\sigma^2}\right]\exp(-\frac{x^2+y^2}{2\sigma^2}) \tag{11-6}$$

式（11-6）称为高斯-拉普拉斯算子（Laplacian of a Gaussian），简称LoG算子，也称为Marr边缘检测算子。

应用 LoG算子时，高斯函数中标准差参数σ的选择很关键，对图像边缘检测效果有很大的影响，对于不同图像应选择不同参数。

LoG算子克服了拉普拉斯算子抗噪声能力比较差的缺点，但是在抑制噪声的同时也可能将原有的比较尖锐的边缘也平滑掉了，造成这些尖锐边缘无法检被测到。

常用的LoG算子是5×5的模板，如图11.4所示。

$$\begin{bmatrix} 0 & 0 & -1 & 0 & 0 \\ 0 & -1 & -2 & -1 & 0 \\ -1 & -2 & 16 & -2 & -1 \\ 0 & -1 & -2 & -1 & 0 \\ 0 & 0 & -1 & 0 & 0 \end{bmatrix}$$

▲图11.4 LoG算子

在5.5节曾指出拉普拉斯算子的响应会产生双边缘，这是复杂分割中所不希望的结果，解决的方法是利用它对阶跃性边缘的零交叉性质来定位边缘（参见5.5.4小节）。

3．Canny边缘检测算子

前面介绍的几种都是基于微分方法的边缘检测算法，它们都只有在图像不含噪声或者首先通过平滑去除噪声的前提下才能正常应用。

在图像边缘检测中，抑制噪声和边缘精确定位是无法同时满足的，一些边缘检测算法通过平滑滤波去除噪声的同时，也增加了边缘定位的不确定性；而提高边缘检测算子对边缘的敏感性的同时，也提高了对噪声的敏感性。Canny算子力图在抗噪声干扰和精确定位之间寻求最佳折衷方案。

Canny对边缘检测质量进行分析，提出3个准则。

（1）信噪比准则：对边缘的错误检测率要尽可能低，尽可能检测出图像的真实边缘，且尽可能减少检测出虚假边缘，获得一个好的结果。在数学上，就是使信噪比SNR尽量大。输出信噪比越大，错误率越小。

$$\mathrm{SNR}=\frac{\left|\int_{-\omega}^{+\omega}G(-x)f(x)\mathrm{d}x\right|}{n_0\left[\int_{-\omega}^{+\omega}f^2(x)\mathrm{d}x\right]^{\frac{1}{2}}} \qquad (11\text{-}7)$$

式中：f(x)是边界为[-ω，ω]有限滤波器的脉冲响应；G(x)代表边缘；n_0是高斯噪声的均方根。

（2）定位精度准则：检测出的边缘要尽可能接近真实边缘。数学上就是寻求滤波函数f(x)使式（11-8）中的Loc尽量大。

$$\mathrm{Loc}=\frac{\left|\int_{-\omega}^{+\omega}G'(-x)f'(x)\mathrm{d}x\right|}{n_0\left[\int_{-\omega}^{+\omega}f'^2(x)\mathrm{d}x\right]^{\frac{1}{2}}} \tag{11-8}$$

式中：G′(−x)、 f′(x)分别是G(-x)、f(x)的一阶导数。

（3）单边缘响应准则：对同一边缘要有低的响应次数，即对单边缘最好只有一个响应。滤波器对边缘响应的极大值之间的平均距离为

$$d_{\max}=2\pi\left[\frac{\int_{-\omega}^{+\omega}f'^2(x)\mathrm{d}x}{\int_{-\omega}^{+\omega}f''^2(x)\mathrm{d}x}\right]^{\frac{1}{2}}\approx kW \tag{11-9}$$

因此在2W宽度内，极大值的数目为

$$N=\frac{2W}{kW}=\frac{2}{k} \tag{11-10}$$

显然，只要固定了k，就固定了极大值的个数。

有了这3个准则，寻找最优的滤波器的问题就转化为泛函的约束优化问题了，问题的解可以由高斯的一阶导数去逼近。

Canny边缘检测的基本思想就是首先对图像选择一定的Gauss滤波器进行平滑滤波，然后采用非极值抑制技术进行处理得到最后的边缘图像。其步骤如下。

（1）用高斯滤波器平滑图像。

这里，利用一个省略系数的高斯函数H(x, y)

$$H(x,y)=\exp(-\frac{x^2+y^2}{2\sigma^2}) \qquad (11\text{-}11)$$

$$G(x,y)=f(x,y)*H(x,y) \qquad (11\text{-}12)$$

其中f(x, y)是图像数据。

（2）用一阶偏导的有限差分来计算梯度的幅值和方向。

利用一阶差分卷积模板：

$$H_1=\begin{vmatrix}-1 & -1\\ 1 & 1\end{vmatrix} \qquad H_2=\begin{vmatrix}1 & -1\\ 1 & -1\end{vmatrix}$$

$$\varphi_1(x,y)=f(x,y)*H_1(x,y) \qquad \varphi_2(x,y)=f(x,y)*H_2(x,y)$$

得到

幅值

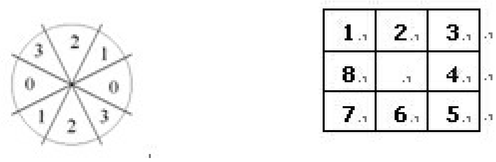
$$\varphi(x,y)=\sqrt{\varphi_1{}^2(x,y)+\varphi_2{}^2(x,y)} \qquad (11\text{-}13)$$

方向

$$\theta_\varphi=\tan^{-1}\frac{\varphi_2(x,y)}{\varphi_1(x,y)} \qquad (11\text{-}14)$$

（3）对梯度幅值进行非极大值抑制。

仅仅得到全局的梯度并不足以确定边缘，为确定边缘，必须保留局部梯度最大的点，而抑制非极大值，即将非局部极大值点置零以得到细化的边缘。



▲图11.5 非极大值抑制

如图11.5所示，4个扇区的标号为0到3，对应3×3邻域的4种可能组合。

在每一点上，邻域的中心像素M与沿着梯度线的两个像素相比。如果M的梯度值不比沿梯度线的两个相邻像素梯度值大，则令M=0。

（4）用双阈值算法检测和连接边缘。

使用两个阈值t_1和t_2(t_1<t_2)，从而可以得到两个阈值边缘图像$n_1[i, j]$和$n_2[i, j]$。由于$n_2[i, j]$使用高阈值得到，因而含有很少的假边缘，但有间断（不闭合）。双阈值法要在$n_2[i, j]$中把边缘连接成轮廓，当到达轮廓的端点时，该算法就在$n_1[i, j]$的8邻点位置寻找可以连接到轮廓上的边缘，这样，算法不断地在 $n_1[i, j]$中收集边缘，直到将 $n_2[i, j]$连接起来为止。t_2用来找到每条线段，t_1用来在这些线段的两个方向上延伸寻找边缘的断裂处，并连接这些边缘。

11.2.3 Matlab实现

Matlab的IPT函数edge可以方便地实现11.2.2小节的几种边缘检测方法，该函数的作用是检测灰度图像中的边缘，并返回一个带有边缘信息的二值图像，其中黑色表示背景，白色表示原图像中的边缘部分。

1．基于梯度算子的边缘检测

使用edge函数进行基于梯度算子的边缘检测的调用语法为

BW = edge(I,type,thresh,direction,'nothinning')

参数说明

● I是需要检测边缘的输入图像。

● type表示梯度算子的种类，合法取值如表11.1所示。

表11.1

合法取值	梯度算子
'sobel'	sobel 算子
'prewitt'	prewitt 算子
'roberts'	robert 算子

● thresh是敏感度阈值参数，任何灰度值低于此阈值的边缘将不会被检测到。其默认值是空矩阵[]，此时算法会自动计算阈值。

● direction指定了我们感兴趣的边缘方向，edge函数将只检测direction中指定方向的边缘。其合法取值如表11.2所示。

表11.2

合法取值	边缘方向
'horizontal'	水平方向
'vertical'	竖直方向
'both'	所有方向

● 可选参数’nothinning’，指定时可以通过跳过边缘细化算法来加快算法运行的速度。在默认时候，这个参数是'thinning'，即进行边缘细化。

返回值

● BW为返回的二值图像，其中0（黑色）为背景，1（白色）为边缘部分。

2．基于高斯-拉普拉斯算子的边缘检测

使用edge函数进行基于高斯-拉普拉斯算子的边缘检测的调用语法为

BW = edge(I,'log',thresh,sigma)

参数说明

● I是待处理的图像。

● 第2个参数为'log'表示采用高斯-拉普拉斯算子。

● thresh是敏感度阈值参数，任何灰度值低于此阈值的边缘将不会被检测到。其默认值是空矩阵[]，此时算法会自动计算阈值。如果将thresh设为0，则输出的边缘图像将包含围绕所有物体的闭合的轮廓线，因为这样的运算会包括输入图像中所有的过零点。

● sigma指定生成高斯滤波器所使用的标准差。默认时，标准差值为2。滤镜大小n×n，n的计算方法为n=ceil(sigma*3)*2+1。

返回值

● BW为返回的二值图像，其中0（黑色）为背景，1（白色）为边缘部分。

3．基于**Canny**算子的边缘检测

BW = edge(I,'canny',thresh,sigma)

参数说明

● I是待处理的图像。

● 第2个参数为'canny'表示采用canny算子。

● thresh 是敏感度阈值参数，其默认值是空矩阵[]。与前面的算法不同，Canny 算法的敏感度阈值是一个列向量，因为需要为算法指定阈值的上下限。在指定阈值矩阵时，第1个元素是阈值下限，第2个元素为阈值上限。如果只指定一个阈值元素，则这个直接指定的值会被作为阈值上限，而它与0.4的积会被作为阈值下限。如果阈值参数没有指定，算法会自行确定敏感度阈值的上、下限。

● sigma指定生成平滑使用的高斯滤波器的标准差。默认时，标准差值为1。滤镜大小n×n，n的计算方法为n=ceil(sigma*3)*2+1。

返回值

● BW为返回的二值图像，其中0（黑色）为背景，1（白色）为边

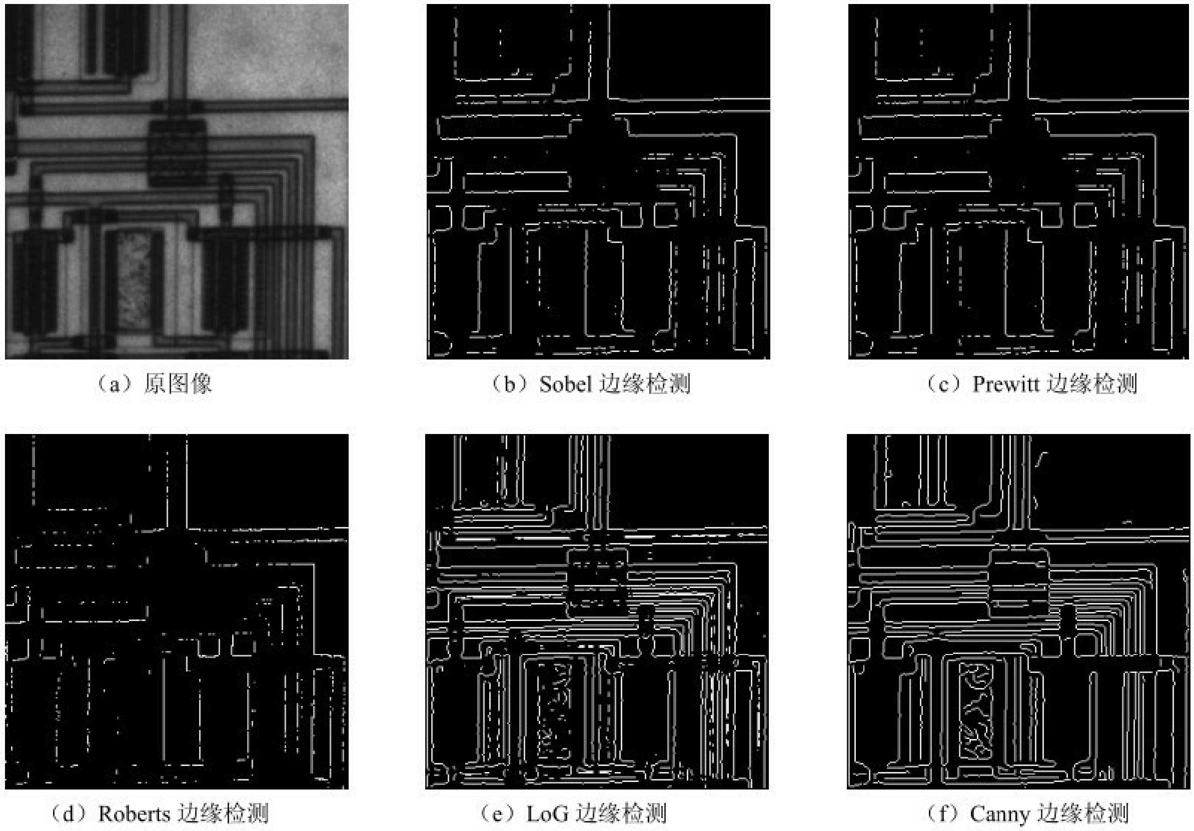
缘部分。

[例11.1]对同一幅图像分别使用上述6种边缘检测算法进行处理，使用算法的默认参数。将输出结果显示在同一窗口中以便进行比较。

相应的Matlab实现代码如下。

```
intensity = imread('circuit.tif'); %读入源图像
bw1 = edge(intensity, 'sobel');
bw2 = edge(intensity, 'prewitt');
bw3 = edge(intensity, 'roberts');
bw4 = edge(intensity, 'log');
bw5 = edge(intensity, 'canny');
subplot(3,2,1); imshow(intensity); title('a'); % 图11.6（a）
subplot(3,2,2); imshow(bw1); title('b'); % 图11.6（b）
subplot(3,2,3); imshow(bw2); title('c'); % 图11.6（c）
subplot(3,2,4); imshow(bw3); title('d'); % 图11.6（d）
subplot(3,2,5); imshow(bw4); title('e'); % 图11.6（e）
subplot(3,2,6); imshow(bw5); title('f'); % 图11.6（f）
```

上述程序的运行结果如图11.6所示。



▲图11.6 几种算法的结果比较

从图11.6中可以看出，不同算法得到的结果存在很大差异，下面进行简要的分析。

● 从边缘定位的精度看

Roberts算子和Log算子定位精度较高。

Roberts算子简单直观，Log算子则利用二阶导数零交叉特性检测边缘。但Log算子只能获得边缘位置信息，不能得到边缘的方向等信息。

● 从对不同方向边缘的响应看

从对边缘方向的敏感性而言，Sobel 算子、Prewitt 算子检测斜向阶跃边缘效果较好，Roberts算子检测水平和垂直边缘效果较好。Log算子则不具备边缘方向检测能力。

Sobel算子可以提供最精确的边缘方向估计。

● 从去噪能力看

Roberts和Log算子定位精度虽然较高，但受噪声影响大。

Sobel算子和Prewitt算子模板相对较大因而去噪能力较强，具有平滑作用，能滤除一些噪声，去掉部分伪边缘，但同时也平滑了真正的边缘，这也正是其定位精度不高的原因。

从总体效果来衡量，Canny算子给出了一种边缘定位精确性和抗噪声干扰性的较好折中。

注意

以上验证结果及分析是基于阶跃变化假设进行的。但真实的灰度变化不一定都是阶跃的，有可能发生在很宽的灰度范围上，且存在灰度的起落。因此，我们应当根据工程实际对各种算子作以比较后加以选用。

11.3 霍夫变换

11.2节介绍了一些边缘检测的有效方法。但实际中由于噪声和光照不均等因素，使得在很多情况下所获得的边缘点是不连续的，必须通过边缘连接将它们转换为有意义的边缘。一般的做法是对经过边缘检测的图像进一步使用连接技术，从而将边缘像素组合成完整的边缘。

霍夫（Hough）变换是一个非常重要的检测间断点边界形状的方法。它通过将图像坐标空间变换到参数空间，来实现直线和曲线的拟合。

11.3.1 直线检测

1．直角坐标参数空间

在图像 x−y坐标空间中，经过点(x_i, y_i)的直线表示为

$$y_i = ax_i + b \tag{11-15}$$

式中：参数a为斜率；b为截距。

通过点(x_i, y_i)的直线有无数条，且对应于不同的 a和 b值，它们都满足式（11-15）。

如果将x_i和y_i视为常数，而将原本得参数a和b看作变量，将式（11-15）表示为

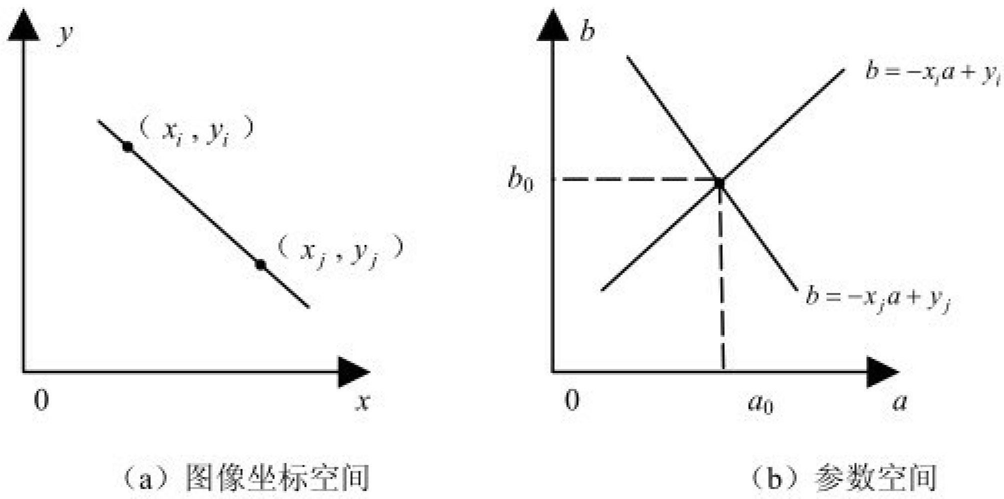
$$b = -x_i a + y_i \tag{11-16}$$

就变换到了参数平面a−b。这个变换就是直角坐标中对于(x_i, y_i)点的Hough变换。该直线是图像坐标空间中的点(x_i, y_i)在参数空间的唯一方程。考虑图像坐标空间中的另一点(x_j, y_j)，它在参数空间中也有相应的一条直线。

$$b = -x_j a + y_j \tag{11-17}$$

这条直线与点(x_i, y_i)在参数空间的直线相交于一点(A_0,b_0)。如图11.7所示。

图像坐标空间中过点(x_i, y_i)和点(x_j, y_j)的直线上的每一点在参数空间a−b上各自对应一条直线，这些直线都相交于点(A_0,b_0)，而A_0、b_0就是图像坐标空间 x−y中点(x_i, y_i)和点(x_j, y_j)所确定的直线的参数。反之，在参数空间相交于同一点的所有直线，在图像坐标空间都有共线的点与之对应。根据这个特性，给定图像坐标空间的一些边缘点，就可以通过Hough变换确定连接这些点的直线方程。



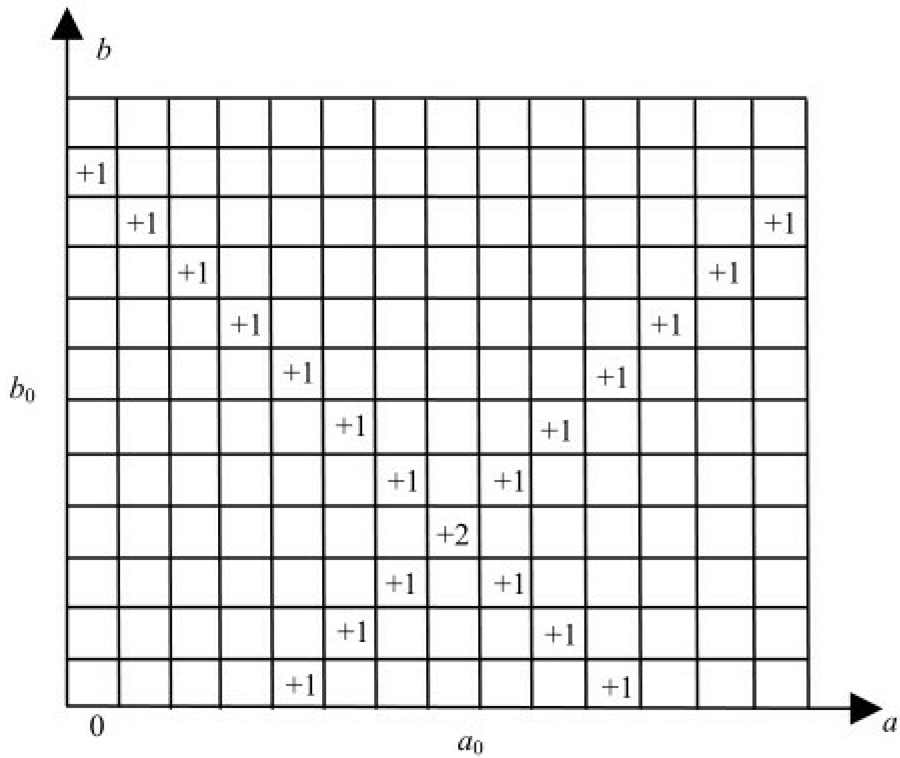
▲图11.7 直角坐标中的Hough变换

具体计算时，可将参数空间视为离散的。建立一个二维累加数组A(a, b)，第一维的范围是图像坐标空间中直线斜率的可能范围，第二维的范围是图像坐标空间中直线截距的可能范围。开始时A(a, b)初始化为0，然后对图像坐标空间的每一个前景点（x_i,y_i），将参数空间中每一个a的离散值代入式（11-16），从而计算出对应的b值。每计算出一对（a,b），都将对应的数组元素A(a, b)加1，即A(a, b)=A(a, b)+1。所有的计算都结束后，在参数空间表决结果中找到 A(a, b)的最大峰值，所对应的A_0、b_0就是原图像中共线点数目最多（共A(A_0,b_0)个共线点）的直线方程的参数；接下来可以继续寻找次峰值和第3第4峰值等，它们对应于原图中共线点数目略少一些的直线。

注意

由于原图中的直线往往具有一定宽度，实际上相当于多条参数极其接近的单像素宽直线，往往对应于参数空间中相邻的多个累加器单元。因此每找到一个当前最大的峰值点后，需要将该点及其附近点清零，以防算法检测出多条极其邻近的“假”直线。

对于图11.7（a）的Hough变换参数空间情况如图11.8所示。



▲图11.8 参数空间表决结果

这种利用二维累加器的离散化方法大大简化了Hough变换的计算，参数空间a-b上的细分程度决定了最终找到直线上点的共线精度。上述的二维累加数组A组也常常被称为Hough矩阵。

注意

使用直角坐标表示直线，当直线为一条垂直直线或者接近垂直直线时，该直线的斜率为无限大或者接近无限大，从而无法在参数空间a-b中表示出来。为了解决这一问题，可以采用极坐标系。

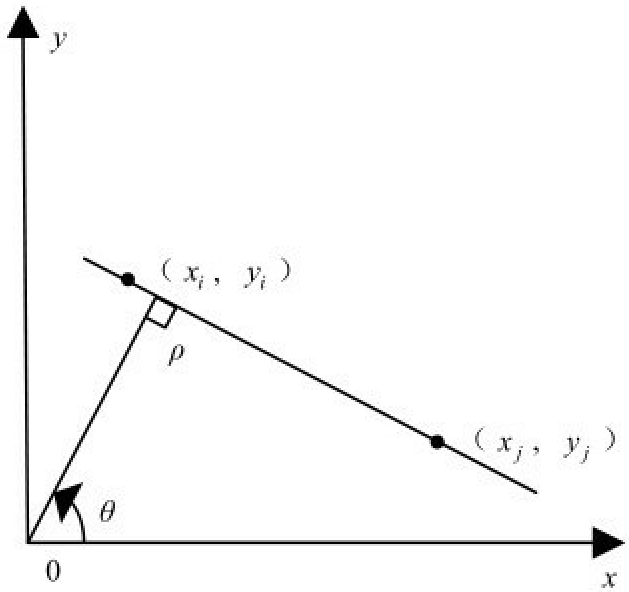
2．极坐标参数空间

极坐标中用如下参数方程表示一条直线。

$$\rho = x\cos\theta + y\sin\theta \tag{11-18}$$

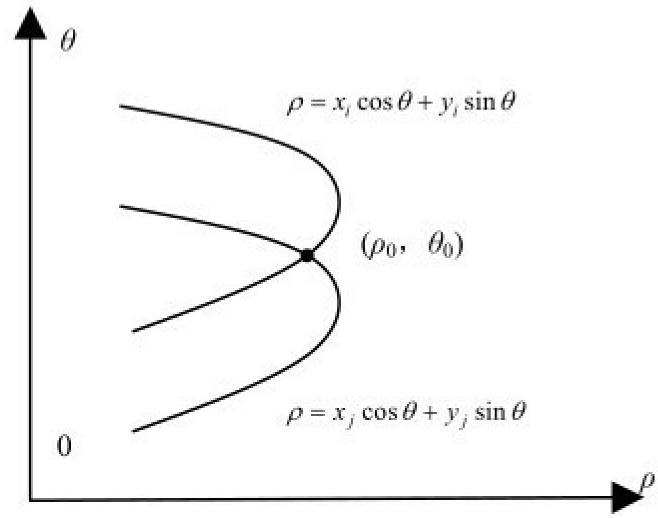
式中：ρ代表直线到原点的垂直距离；θ代表x轴到直线垂线的角

度，取值范围为±90°，如图11.9所示。



▲图11.9 直线的参数式表示

与直角坐标类似，极坐标中的Hough变换也将图像坐标空间中的点变换到参数空间中。在极坐标表示下，图像坐标空间中共线的点变换到参数空间中后，在参数空间都相交于同一点，此时所得到的 ρ、θ 即为所求的直线的极坐标参数。与直角坐标不同的是，用极坐标表示时，图像坐标空间的共线的两点(x_i, y_i)和(x_j, y_j)映射到参数空间是两条正弦曲线，相交于点$(\rho_0，\theta_0)$，如图 11.10所示。



▲图11.10 笛卡儿坐标映射到参数空间

具体计算时，与直角坐标类似，也要在参数空间中建立一个二维数组累加器A，只是取值范围不同。对一幅大小为D×D的图像，通常ρ的取值范围为[$-\sqrt{2}D/2$，$\sqrt{2}D/2$]，θ的取值范围为[−90°，90°]。计算方法与直角坐标系中累加器的计算方法相同，最后得到的最大的A所对应的(ρ，θ)。

11.3.2 曲线检测

Hough变换同样适用于方程已知的曲线检测。图像坐标空间中的一条已知的曲线方程也可以建立其相应的参数空间。由此，图像坐标空间中的一点，在参数空间中就可以映射为相应的轨迹曲线或者曲面。若参数空间中对应各个间断点的曲线或者曲面能够相交，就能够找到参数空间的极大值以及对应的参数；若参数空间中对应各个间断点的曲线或者曲面不能相交，则说明间断点不符合某已知曲线。

Hough变换做曲线检测时，最重要的是写出图像坐标空间到参数空间的变换公式。例如，对于已知的圆方程，其直角坐标的一般方程为

$$(x-a)^2+(y-b)^2=r^2 \tag{11-19}$$

其中(a，b)为圆心坐标，r为圆的半径，它们为图像的参数。那么，参数空间可以表示为(a，b， r)，图像坐标空间中的一个圆对应参数空间中的一点。

具体计算时，与前面讨论的方法相同，只是数组累加器为三维A(a，b，r)。计算过程是让a、b在取值范围内增加，解出满足上式的r值，每计算出一个(a，b，r)值，就对数组元素A(a，b，r)加1。计算结束后，找到的最大的A(a，b，r)所对应的a、b、r就是所求的圆的参数。

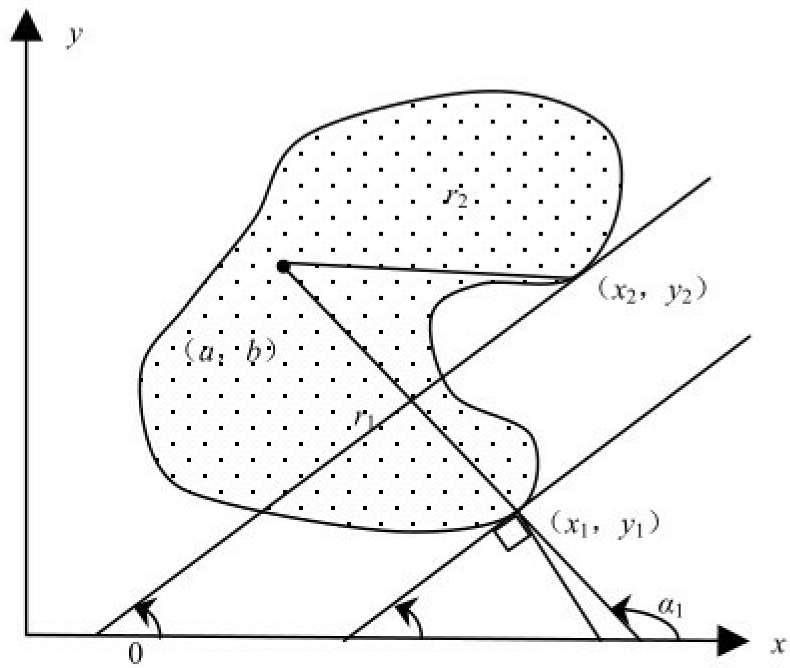
与直线检测一样，曲线检测也可以通过极坐标形式计算。

注意

通过Hough变换做曲线检测时，参数空间的大小将随着参数个数的增加呈指数上升的趋势，故在实际使用时要尽量减少描述曲线的参数数目。因此，这种曲线检测的方法只对检测参数较少的曲线有意义。

11.3.3 任意形状的检测

这里所说的任意形状的检测，是指应用广义 Hough 变换去检测某一任意形状边界的图形。它首先选取该形状中的任意点(a，b)为参考点，然后从该任意形状图形的边缘上每一点计算其切线方向φ和到参考点(a，b)位置偏移矢量r，以及r与x轴的夹角α，如图11.11所示。



▲图11.11 广义Hough变换

参考点(a，b)的位置可由下式算出。

$$a = x + r(\phi)\cos(\alpha(\phi)) \tag{11-20}$$

$$b = x + r(\phi)\sin(\alpha(\phi)) \tag{11-21}$$

利用广义Hough变换检测任意形状边界的主要步骤如下。

（1）在预知区域形状的条件下，将物体边缘形状编成参考表。对于每个边缘点计算梯度角ϕ_i，对每一个梯度角ϕ_i，算出对应于参考点的距离R_i和角度α_i。例如，图11.11中，同一个梯度角φ对应两个点，则参考表表示为

$$\phi\text{：}(r_1,\ \alpha_1)\ (r_2,\ \alpha_2)$$

同理，可以表示出其他梯度角ϕ_i所对应的参考表。

（2）在参数空间建立一个二维累加数组 A(a，b)，初值为0。对边

缘上的每一点，计算出该点处的梯度角，然后，由上式计算出每一个可能的参考点的位置值，对相应的数组元素A(a，b)加1。

（3）计算结束后，具有最大值的数组元素A(a，b)所对应的a、b值即为图像坐标空间中所求的参考点。

求出参考点后，整个目标的边界就可以确定了。

11.3.4 Hough变换直线检测的Matlab实现

通过Hough变换在二值图像中检测直线需要3个步骤。

（1）利用hough()函数执行霍夫变换，得到霍夫矩阵。

（2）利用houghpeaks()函数在霍夫矩阵中寻找峰值点。

（3）利用houghlines()函数在之前2步结果的基础上得到原二值图像中的直线信息。

1．霍夫变换——hough

hough函数对一幅二值图像执行Hough变换，得到Hough矩阵。调用形式为

[H, theta, rho] = hough(BW,param1,val1,param2,val2)

参数说明

● BW是边缘检测后的二值图像。

● 可选参数对param1、value1以及param2、value2的合法取值如表11.3所示。

表11.3 param参数的合法取值及其意义

param 取值	含　义
'ThetaResolution'	Hough 矩阵中 θ 轴方向上单位区间的长度（以“度”为单位），可取（0，90）区间上的实数，默认为 1
'RhoResolution'	Hough 矩阵中 ρ 轴方向上单位区间的长度，可取（0，norm(size(BW))）区间上的实数，默认为 1

返回值

● H是变换得到的Hough矩阵。

● theta和rho为分别对应于Hough矩阵每一列和每一行的θ和ρ值组成的向量。

2．寻找峰值**——houghpeaks**

houghpeaks函数用于在Hough矩阵中寻找指定数目的峰值点，调用形式如下。

peaks = houghpeaks(H,numpeaks, param1,val1,param2,val2)

参数说明

● H是由hough()函数得到的Hough矩阵。

● numpeaks是要寻找的峰值数目，默认为1。

● 可选参数对儿param1、value1以及param2、value2的合法取值如表11.4所示。

表11.4 param参数的合法取值及其意义

param 取值	含　义
'Threshold'	峰值的阈值，只有大于该阈值的点才被认为是可能的峰值，其取值>0，默认为 0.5*max(*H*(:))
'NHoodSize'	在每次检测出一个峰值后，NhoodSize 指出了在该峰值周围需要清零的邻阈信息。以向量[*M N*]的形式给出，其中 *M*、*N* 均为正的奇数。默认为大于等于 size(*H*)/50 的最小奇数

返回值

● peaks是一个Q×2的矩阵，每行的2个元素分别为某一峰值点在Hough矩阵中的行、列索引，Q为找到的峰值点的数目。

3．提取直线段**——houghlines**

Houghlines函数根据Hough矩阵的峰值检测结果提取直线段，调用语法为

lines = houghlines(BW,theta, rho, peaks, param1, val1, param2, val2)

参数说明

● BW是边缘检测后的二值图像。

● theta和rho是Hough矩阵每一列和每一行的θ和ρ值组成的向量，由hough函数返回。

● peaks是一个包含峰值点信息的Q×2的矩阵，由houghpeaks函数返回。

● 可选参数对儿param1、value1以及param2、value2的合法取值如表11.5所示。

表11.5 param参数的合法取值及其意义

param 取值	含　义
'FillGap'	线段合并的阈值：如果对应于Hough矩阵某一个单元格（相同的θ和ρ）的2个线段之间的距离小于FillGap，则合并为1个直线段。默认值为20
'MinLength'	检测的直线段的最小长度阈值：如果检测出的直线段长度大于MinLength，则保留；丢弃所有长度小于MinLength的直线段。默认值为40

返回值

● lines是一个结构体数组，数组长度是找到的直线条数，而每一个数组元素（直线段结构体）的内部结构如表11.6所示。

表11.6 lines的结构

域	含　义
point1	直线段的端点1
point2	直线段的端点2
theta	对应在霍夫矩阵中的θ
rho	对应在霍夫矩阵中的ρ

[例11.2]利用Hough变换对Matlab示例图片circuit.tif进行直线检测，显示Hough矩阵和检测到的峰值，并在原图中标出符合要求的所有直线段。

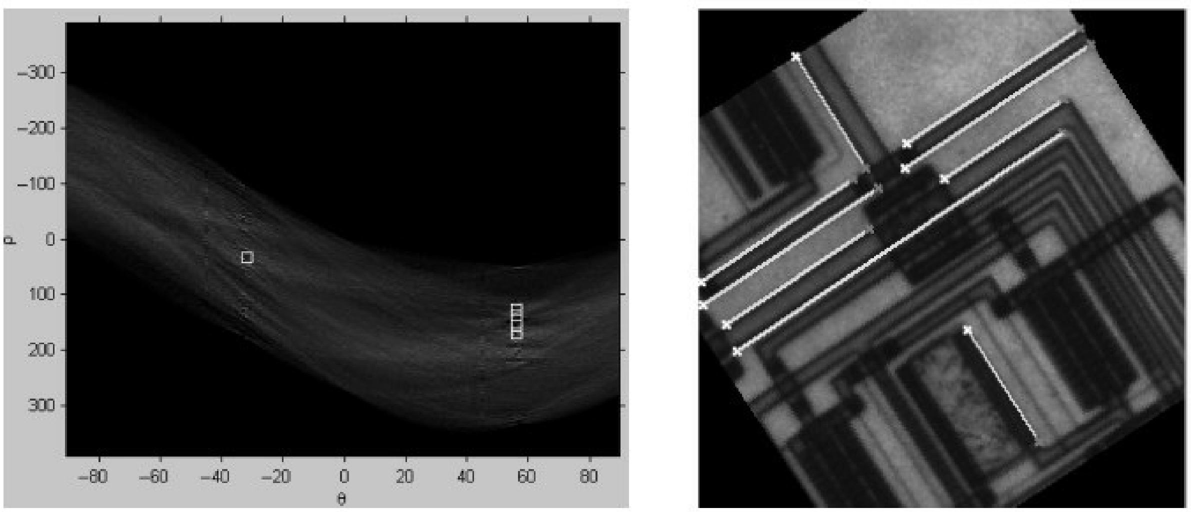
相应的Matlab实现代码如下。

```
I = imread('circuit.tif'); % 读取图像
```

```
% 旋转图像并寻找边缘
rotI = imrotate(I,33,'crop');
BW = edge(rotI,'canny');
% 执行Hough变换并显示Hough矩阵
[H,T,R] = hough(BW);
imshow(H,[],'XData',T,'YData',R,'InitialMagnification','fit'); %得到图
11.12（a）
xlabel('\theta'), ylabel('\rho');
axis on, axis normal, hold on;
% 在Hough矩阵中寻找前5个大于Hough矩阵中最大值0.3倍的峰值
P = houghpeaks(H,5,'threshold',ceil(0.3*max(H(:))));
x = T(P(:,2)); y = R(P(:,1)); % 由行、列索引转换成实际坐标
plot(x,y,'s','color','white'); % 在Hough矩阵图像中标出峰值位置
% 找到并绘制直线
lines = houghlines(BW,T,R,P,'FillGap',5,'MinLength',7); % 合并距离小
于5的线段，丢弃所有长度
小于7的直线段
figure, imshow(rotI), hold on
max_len = 0;
for k = 1:length(lines) % 依次标出各条直线段
  xy = [lines(k).point1; lines(k).point2];
  plot(xy(:,1),xy(:,2),'LineWidth',2,'Color','green');
  % 绘制线段端点
  plot(xy(1,1),xy(1,2),'x','LineWidth',2,'Color','yellow');
  plot(xy(2,1),xy(2,2),'x','LineWidth',2,'Color','red');
  % 确定最长的线段
  len = norm(lines(k).point1 - lines(k).point2);
```

```
    if ( len > max_len)
      max_len = len;
      xy_long = xy;
    end
  end
  % 高亮显示最长线段
  plot(xy_long(:,1),xy_long(:,2),'LineWidth',2,'Color','cyan'); %得到图11.12（b）
```

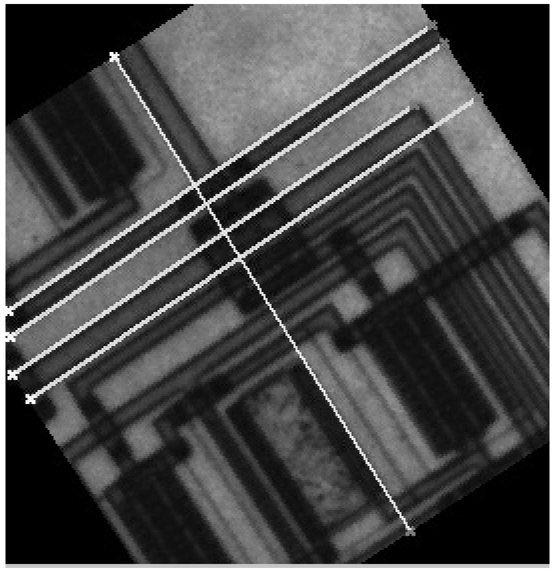
上述程序的运行结果如图11.12所示。



（a）Hough 矩阵和峰值点　　（b）检测出的直线段

▲图11.12 Hough变换效果（'FillGap'=5）

注意到在houghpeaks函数执行后，我们共得到了5个峰值，然而图11.12（b）的结果中却出现了8条直线段，这正是 houghlines函数中'FillGap'参数的作用。将'FillGap'设定为 80可以合并原本共线（有相同的θ和ρ）的各个直线段，将得到如图11.13所示的结果。



▲图11.13 检测出的直线段（'FillGap'=80）

注意

霍夫变换只能处理二值图像，一般在执行变换前需要首先在图像上执行边缘检测。

在金羽图书论坛（http://bbs.book95.com）的“金羽图书与答疑”板块与本书同名的主题帖子附件的第11章目录中我们提供了一副名为line.bmp的示例图片，供读者检验Hough变换的效果。

11.4 阈值分割

我们曾在3.5节学习过灰度阈值变换的相关知识，利用灰度阈值变换分割图像就称为阈值分割，它是一种基本的图像分割方法。

阈值分割的基本思想是确定一个阈值，然后把每个像素点的灰度值和阈值相比较，根据比较的结果把该像素划分为两类——前景或者背景。一般来说，阈值分割可以分成以下3步。

（1）确定阈值。

（2）将阈值和像素比较。

（3）把像素归类。

其中第（1）步阈值的确定是最重要的。阈值的选择将直接影响分割的准确性以及由此产生的图像描述、分析的正确性。

11.4.1 阈值分割方法

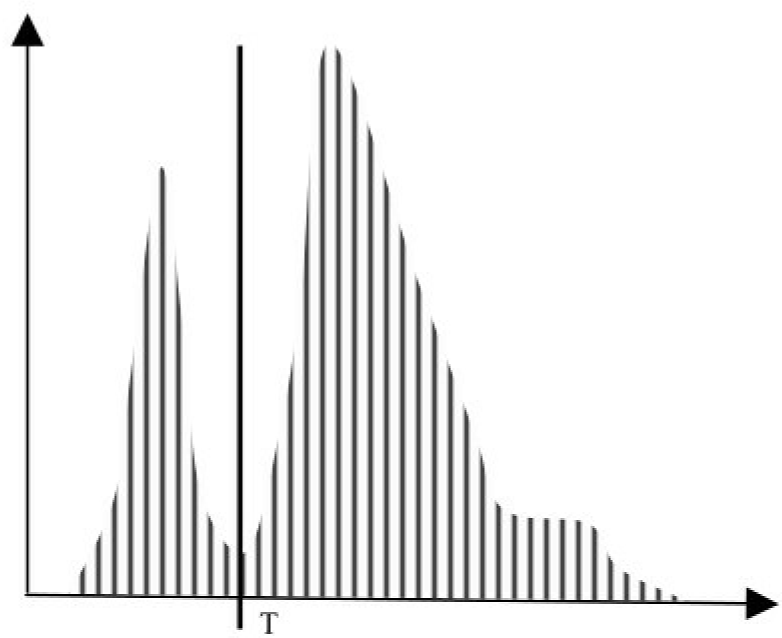
阈值分割常用的方法一般有以下几种。

1．实验法

实验法是通过人眼的观察，对已知某些特征的图像，只要试验不同的阈值，然后看是否满足已知特征即可。这种方法的问题是适用范围窄，使用前必须事先知道图像的某些特征，如平均灰度等，而且分割后的图像质量的好坏受主观局限性很大。

2．根据直方图谷底确定阈值

如果图像的前景物体内部和背景区域的灰度值分布都比较均匀，那么这个图像的灰度直方图将具有明显双峰，此时可以选择两峰之间的谷底作为阈值，如图11.14所示。



▲图11.14 根据直方图谷底确定阈值

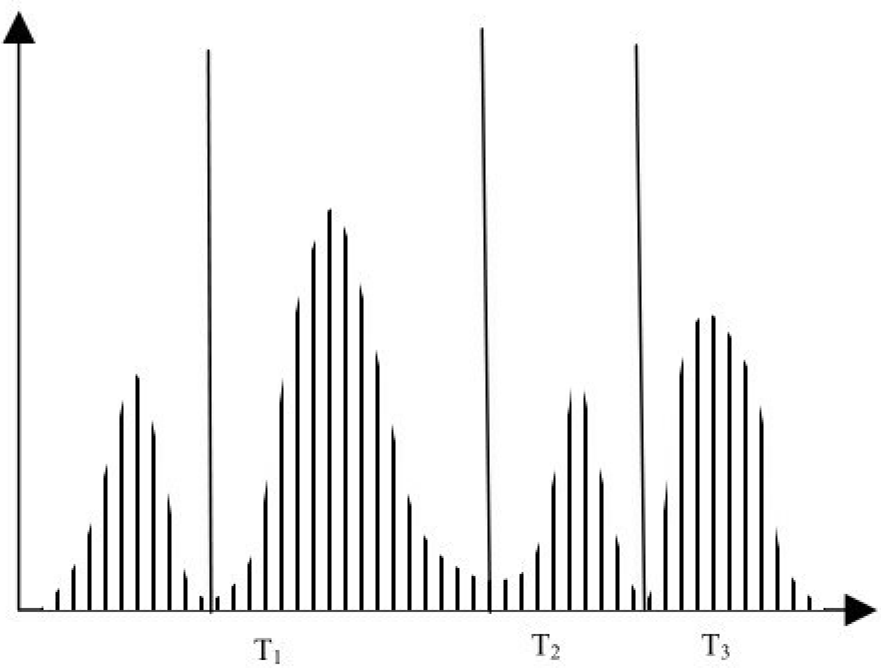
其表达式为

$$g(x)=\begin{cases}255 & f(x,y)\geqslant T\\ 0 & f(x,y)<T\end{cases} \quad (11\text{-}22)$$

式中：g(x)为阈值运算后的二值图像。

此种单阈值分割方法简单易操作，但是当两个峰值相差很远时不适用，而且，此种方法比较容易受到噪声的影响，进而导致阈值选取的误差。

对于有多个峰值的直方图，可以选取多个阈值，这些阈值的选取一般没有统一的规则，要根据实际情况运用，如图11.15所示。



▲图11.15 多峰值直方图确定阈值

注意

由于直方图是各灰度的像素统计，其峰值和谷底特性不一定代表目标和背景。因此，如果没有图像其他方面的知识，只靠直方图进行图像分割是不一定准确的。

3．迭代选择阈值法

迭代式阈值选择方法的基本思想是开始选择一个阈值作为初始估计值，然后按照某种规则不断地更新这一估计值，直到满足给定的条件为止。这个过程的关键是选择怎么样的迭代规则。一个好的迭代规则必须既能够快速收敛，又能够在每一个迭代过程中产生的优于上次迭代的结果。下面是一种迭代选择阈值算法。

（1）选择一个T的初始估计值。

（2）利用阈值T把图像分为两个区域R_1和R_2。

（3）对区域R_1和R_2中的所有像素计算平均灰度值μ_1和μ_2。

（4）计算新的阈值：

$$T=\frac{1}{2}(\mu_1+\mu_2) \tag{11-23}$$

（5）重复步骤（2）～（4），直到逐次迭代所得的T值小于事先定义的参数T。

4．最小均方误差法

最小均方误差法也是常用的阈值分割法之一。这种方法通常以图像中的灰度为模式特征，假设各模式的灰度是独立分布的随机变量，并假设图像中待分割的模式服从一定的概率分布。一般来说，采用的是正态分布，即高斯概率分布。

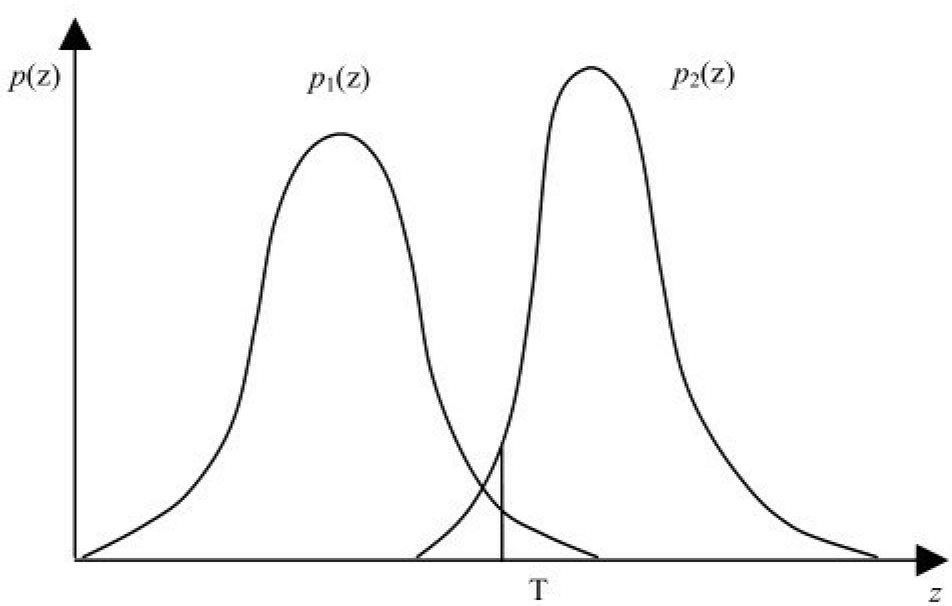
首先假设一幅图像仅包含两个主要的灰度区域——前景和背景。令z表示灰度值，p(z)表示灰度值概率密度函数的估计值。假设概率密度函数一个对应于背景的灰度值，另一个对应于图像中前景即对象的灰度值。则描述图像中整体灰度变换的混合密度函数是

$$p(z) = P_1 p_1(z) + P_2 p_2(z) \tag{11-24}$$

其中p_1是前景中具有值z的像素出现的概率，p_2是背景中具有值z的像素出现的概率，两者的关系为

$$P_1 + P_2 = 1 \tag{11-25}$$

即图像中的像素只能是属于前景或者背景，没有第3种情况。现要选定一个阈值T，将图像上的像素进行归类。采用最小均方误差法的目的是选择T时，使对一个给定像素进行是前景还是背景的分类时出错的概率最小，如图11.16所示。



▲图11.16 最小均方误差法确定阈值

当选定阈值T时，将一个背景点当成前景点进行分类错误出现的概率为

$$E_1(T) = \int_{-\infty}^{T} p_2(z)\mathrm{d}z \tag{11-26}$$

当选定阈值T时，将一个前景点当成背景点进行分类错误出现的概率为

$$E_2(T) = \int_T^{\infty} p_1(z)\mathrm{d}z \tag{11-27}$$

总错误率为

$$E(T) = P_2 E_1(T) + P_1 E_2(T) \tag{11-28}$$

要找到出错最少的阈值T，需要将E(T)对T求微分并令微分式等于0，于是结果是

$$P_1 p_1(T) = P_2 p_2(T) \tag{11-29}$$

根据这个等式解出T，即为最佳阈值。

下面讨论如何得到T的解析式。

要想得到T的分析表达式，则需要已知两个密度概率函数的解析式。一般假设图像的前景和背景的灰度分布都满足正态分布，即使用高斯密度概率函数。此时

$$p_1(z) = \frac{1}{\sqrt{2\pi}\sigma_1}\exp[-\frac{(z-\mu_1)^2}{2\sigma_1{}^2}] \tag{11-30}$$

$$p_2(z) = \frac{1}{\sqrt{2\pi}\sigma_2}\exp[-\frac{(z-\mu_2)^2}{2\sigma_2{}^2}] \tag{11-31}$$

若 $\sigma^2 = \sigma_1{}^2 = \sigma_2{}^2$ ，则单一阈值

$$T = \frac{\mu_1+\mu_2}{2} + \frac{\sigma^2}{\mu_1-\mu_2}\ln\left(\frac{P_2}{P_1}\right) \tag{11-32}$$

若P1=P2=0.5，则最佳阈值是均值的平均数，即位于曲线$p_1(z)$和$p_2(z)$的交点处。

$$T=\frac{\mu_1+\mu_2}{2} \tag{11-33}$$

一般来讲，确定能使均方误差最小的参数很复杂，而上述讨论也在图像的前景和背景都为正态分布的条件下成立。但是，图像的前景和背景是否都为正态分布，也是一个具有挑战性的问题。

5．最大类间方差法

在对图像进行阈值分割时，选定的分割阈值应使前景区域的平均灰度、背景区域的平均灰度与整幅图像的平均灰度之间差别最大，这种差异用区域的方差来表示。由此，Otsu在1978年提出了最大方差法。该算法在判决分析最小二乘法原理的基础上推导得出，计算简单，是一种稳定、常用的算法。

设图像中灰度为i的像素数为n_i，灰度范围为[0, L-1]，则总的像素数为

$$N=\sum_{i=0}^{L-1} n_i \tag{11-34}$$

各灰度值出现概率为

$$p_i=\frac{n_i}{N} \tag{11-35}$$

对于p_i，有

$$\sum_{i=0}^{L-1} p_i=1 \tag{11-36}$$

把图像中的像素用阈值T分成两类c_0和c_1，c_0由灰度值在[0, T-1]的像素组成，c_1由灰度值在[T, L-1]的像素组成。则区域c_0和c_1的概率分别为

$$P_0 = \sum_{i=0}^{T-1} p_i \tag{11-37}$$

$$P_1 = \sum_{i=T}^{L-1} p_i = 1 - P_0 \tag{11-38}$$

区域c_0和c_1的平均灰度分别为

$$\mu_0 = \frac{1}{P_0}\sum_{i=0}^{T-1} ip_i = \frac{\mu(T)}{P_0} \tag{11-39}$$

$$\mu_1 = \frac{1}{P_1}\sum_{i=T}^{L-1} ip_i = \frac{\mu - \mu(T)}{1 - P_0} \tag{11-40}$$

其中，μ是整幅图像的平均灰度。

$$\mu = \sum_{i=0}^{L-1} ip_i = \sum_{i=0}^{t-1} ip_i + \sum_{i=T}^{L-1} ip_i = P_0\mu_0 + P_1\mu_1 \tag{11-41}$$

两个区域的总方差为

$$\sigma_B{}^2 = P_0(\mu_0 - \mu)^2 + P_1(\mu_1 - \mu)^2 = P_0P_1(\mu_0 - \mu_1)^2 \tag{11-42}$$

让T在[0,L-1]范围内依次取值，使 $\sigma_B{}^2$最大的T值便是最佳区域分割阈值。

该方法不需要人为设定其他参数，是一种自动选择阈值的方法，而且能得到较好的结果。它不仅适用于包含两个区域的单阈值选择，也同样适用于多区域的多阈值选择。

11.4.2 Matlab实现

1．最大类间方差法

Matlab中和阈值变换相关的2个主要函数是im2bw和graythresh。实际上，graythresh函数即实现了11.4.1小节介绍的最大类间方差法。它们的用法在3.5.2小节有详细地介绍，这里不再赘述。

2．迭代选择阈值法

下面还给出了根据迭代选择阈值法原理编写的Matlab函数autoThreshold()，它位于金羽图书论坛（http://bbs.book95.com）的"金羽图书与答疑"板块与本书同名的主题帖子附件的"chapter11\code"目录中。

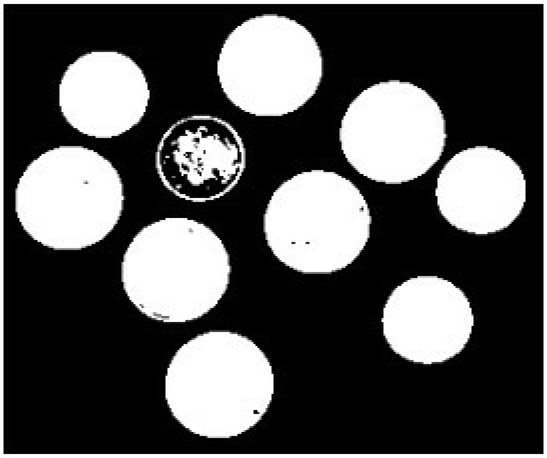
```
function [Ibw, thres] = autoThreshold(I)
% 迭代法自动阈值分割
%
% 输入：I - 要进行自动阈值分割的灰度图像
% 输出：Ibw - 分割后的二值图像
%       thres - 自动分割采用的阈值
thres = 0.5 * (double(min(I(:))) + double(max(I(:)))); %初始阈值
done = false; %结束标志
while ～done
   g = I >= thres;
   Tnext = 0.5 * (mean(I(g)) + mean(I(～g)));
   done = abs(thres - Tnext) < 0.5;
   thres = Tnext;
end;
Ibw = im2bw(I, thres/255); % 二值化
```

下面给出一个利用autoThreshold()函数对Matlab内置图像coins.png进

行自动阈值分割的调用示例。

```
f = imread('coins.png'); %读入原图像
[Ibw, thres] = autoThreshold(I);
imshow(Ibw) %得到图11.17
thres %显示所用阈值
thres =
  126.0522
```

上述程序的运行结果如图11.17所示。



▲图11.17 迭代选择法的自动阈值分割结果

11.5 区域分割

前面所讲的图像分割方法都是基于像素的灰度来进行阈值分割，本节将讨论以区域为基础的图像分割技术。传统的区域分割方法有区域生长和区域分裂与合并，其中最基础的是区域生长法。

11.5.1 区域生长及其实现

区域生长是根据事先定义的准则将像素或者子区域聚合城更大区域

的过程。其基本思想是从一组生长点开始（生长点可以是单个像素，也可以为某个小区域），将与该生长点性质相似的相邻像素或者区域与生长点合并，形成新的生长点，重复此过程直到不能生长为止。生长点和相邻区域的相似性判据可以是灰度值、纹理、颜色等多种图像信息。

1．区域生长算法

区域生长一般有3个步骤。

（1）选择合适的生长点。

（2）确定相似性准则即生长准则。

（3）确定生长停止条件。

一般来说，在无像素或者区域满足加入生长区域的条件时，区域生长就会停止。

图11.18给出一个区域生长的实例：图11.18（a）所示为原图像，数字表示像素的灰度。以灰度为8的像素为初始的生长点，记为f(i, j)。在8邻域内，生长准则是待测点灰度值与生长点灰度值相差为1或者0。那么，如图11.18（b）所示，第一次区域生长后，f(i-1, j)、f(i, j-1)、f(i, j+1)与中心点灰度值相差都为1，因而被合并。第二次生长后，如图11.18（c）所示，f(i+1, j)被合并。第三次生长后，如图11.18（d）所示，f(i+1, j-1)、f(i+2, j)被合并，至此，已经不存在满足生长准则的像素点，生长停止。

$$\begin{bmatrix} 4 & 3 & 7 & 3 & 3 \\ 1 & 7 & (8) & 7 & 5 \\ 0 & 5 & 6 & 1 & 3 \\ 2 & 2 & 6 & 0 & 4 \\ 1 & 2 & 1 & 3 & 1 \end{bmatrix}$$

（a）原图像灰度矩阵生长点

$$\begin{bmatrix} 4 & 3 & (7) & 3 & 3 \\ 1 & (7) & (8) & (7) & 5 \\ 0 & 5 & 6 & 1 & 3 \\ 2 & 2 & 6 & 0 & 4 \\ 1 & 2 & 1 & 3 & 1 \end{bmatrix}$$

（b）第一次区域生长结果

$$\begin{bmatrix} 4 & 3 & (7) & 3 & 3 \\ 1 & (7) & (8) & (7) & 5 \\ 0 & 5 & (6) & 1 & 3 \\ 2 & 2 & 6 & 0 & 4 \\ 1 & 2 & 1 & 3 & 1 \end{bmatrix}$$

（c）第二次区域生长结果

$$\begin{bmatrix} 4 & 3 & (7) & 3 & 3 \\ 1 & (7) & (8) & (7) & 5 \\ 0 & (5) & (6) & 1 & 3 \\ 2 & 2 & (6) & 0 & 4 \\ 1 & 2 & 1 & 3 & 1 \end{bmatrix}$$

（d）第三次区域生长结果

▲图11.18 区域生长示意图

上面的方法比较的是单个像素与其邻域的灰度特征以实现区域生长，也有一种混合型区域生长把图像分割成若干小区域，比较相邻小区域的相似性，如果相似则合并。在实际中，区域生长时经常还要考虑到生长的“历史”，还要根据区域的尺寸、形状等图像的全局性质来决定区域的合并。

2．Matlab实现

下面给出一个基于种子点 8-邻域的区域生长算法的 Matlab 实现，它位于金羽图书论坛（http://bbs.book95.com）的“金羽图书与答疑”板块与本书同名的主题帖子附件的“chapter11\code”目录下的regionGrow.m文件中。

```
function J = regionGrow(I)
% 区域生长，需要以交互方式设定初始种子点，具体方法为鼠标单击图像中一点后，按下回车键
%
```

```
% 输入：I - 原图像
% 输出：J - 输出图像
if isinteger(I)
  I=im2double(I);
end
figure,imshow(I),title('原始图像')
[M,N]=size(I);
[y,x]=getpts;   %获得区域生长起始点
x1=round(x);   %横坐标取整
●1=round(y);   %纵坐标取整
seed=I(x1,y1);   %将生长起始点灰度值存入seed中
J=zeros(M,N);   %作一个全零与原图像等大的图像矩阵J，作为输出
图像矩阵
J(x1,y1)=1;   %将J中与所取点相对应位置的点设置为白
sum=seed;   %储存符合区域生长条件的点的灰度值的和
suit=1;   %储存符合区域生长条件的点的个数
count=1;   %记录每次判断一点周围八点符合条件的新点的数目
threshold=0.15;   %阈值，注意需要和double类型存储的图像相符合
while count>0
  s=0;   %记录判断一点周围八点时，符合条件的新点的灰度值之
和
    count=0;
    for i=1:M
      for j=1:N
        if J(i,j)==1
          if (i-1)>0 & (i+1)<(M+1) & (j-1)>0 & (j+1)<(N+1) %判断此点
是否为图像边界上的点
```

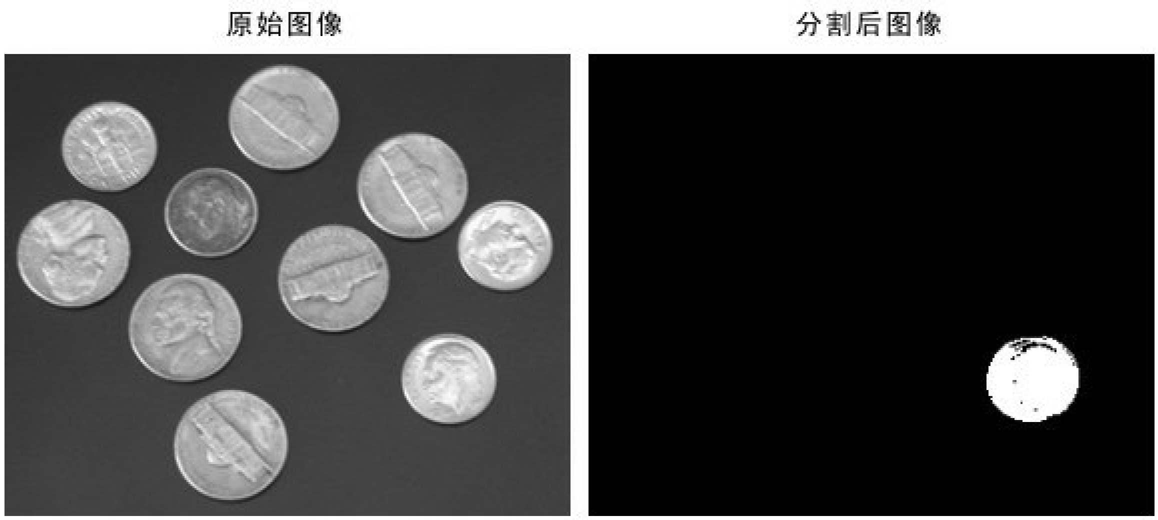
```
            for u= -1:1    %判断点周围八点是否符合阈值条件
            for v= -1:1
           if J(i+u,j+v)==0 & abs(I(i+u,j+v)-seed)<=threshold&
1/(1+1/15*abs(I(i+u,
    j+v)-seed))>0.8
              J(i+u,j+v)=1;
           %判断是否尚未标记，并且为符合阈值条件的点
           %符合以上两条件即将其在J中与之位置对应的点设置为白
          count=count+1;
          s=s+I(i+u,j+v);    %此点的灰度之加入s中
         end
        end
       end
      end
     end
    end
   end
   suit=suit+count;    %将n加入符合点数计数器中
   sum=sum+s;    %将s加入符合点的灰度值总合中
   seed=sum/suit;    %计算新的灰度平均值
  end
```

下面给出一个利用regionGrow()函数对Matlab内置图像coins.png进行基于种子点的区域生长的调用示例。

```
>> I = imread('coins.png');
>> J = regionGrow(I);
>> figure,imshow(J), title('分割后图像')
```

上述程序运行后，会弹出一个包含原图像的窗口，用户可以用鼠标

在其中选取一个种子点并按下回车键，之后会出现分割结果，如图11.19所示。



（a）原始图像　　（b）区域生长结果

▲图11.19 区域生长算法示例

11.5.2 区域分裂与合并及其Matlab实现

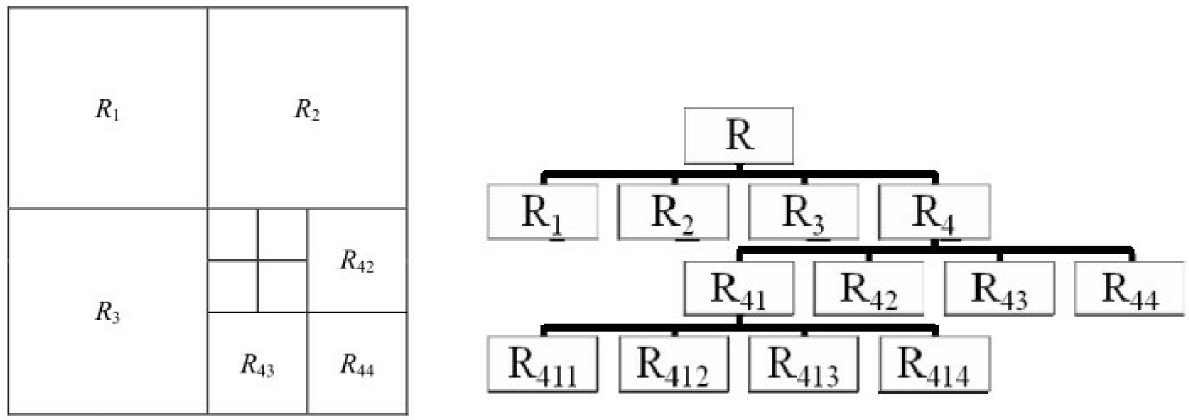
区域生长是从一组生长点开始的，另一种方法是在开始时将图像分割为一系列任意不相交的区域，然后将它们合并或者拆分以满足限制条件，这就是区域分裂与合并。通过分裂，可以将不同特征的区域分离开；而通过合并，将相同特征的区域合并起来。

1． 区域分裂与合并算法

（1）分裂

令 R 表示整个图像区域，P 代表某种相似性准则。一种区域分裂方法是首先将图像等分为 4个区域，然后反复将分割得到的子图像再次分为4个区域，直到对任意R_i，$P(R_i)$=TRUE，表示区域R_i已经满足相似性准则（譬如说该区域内的灰度值相等或相近），此时不再进行分裂操作。如果$P(R_i)$=FALSE，则将R_i分割为4个区域。如此继续下去，直到

$P(R_i)$=TRUE或者已经到单个像素。这个过程可以用四叉树形式表示，如图11.20所示。



（a）被分割的图像　　（b）对应的四叉树

▲图11.20 四叉树算法示意图

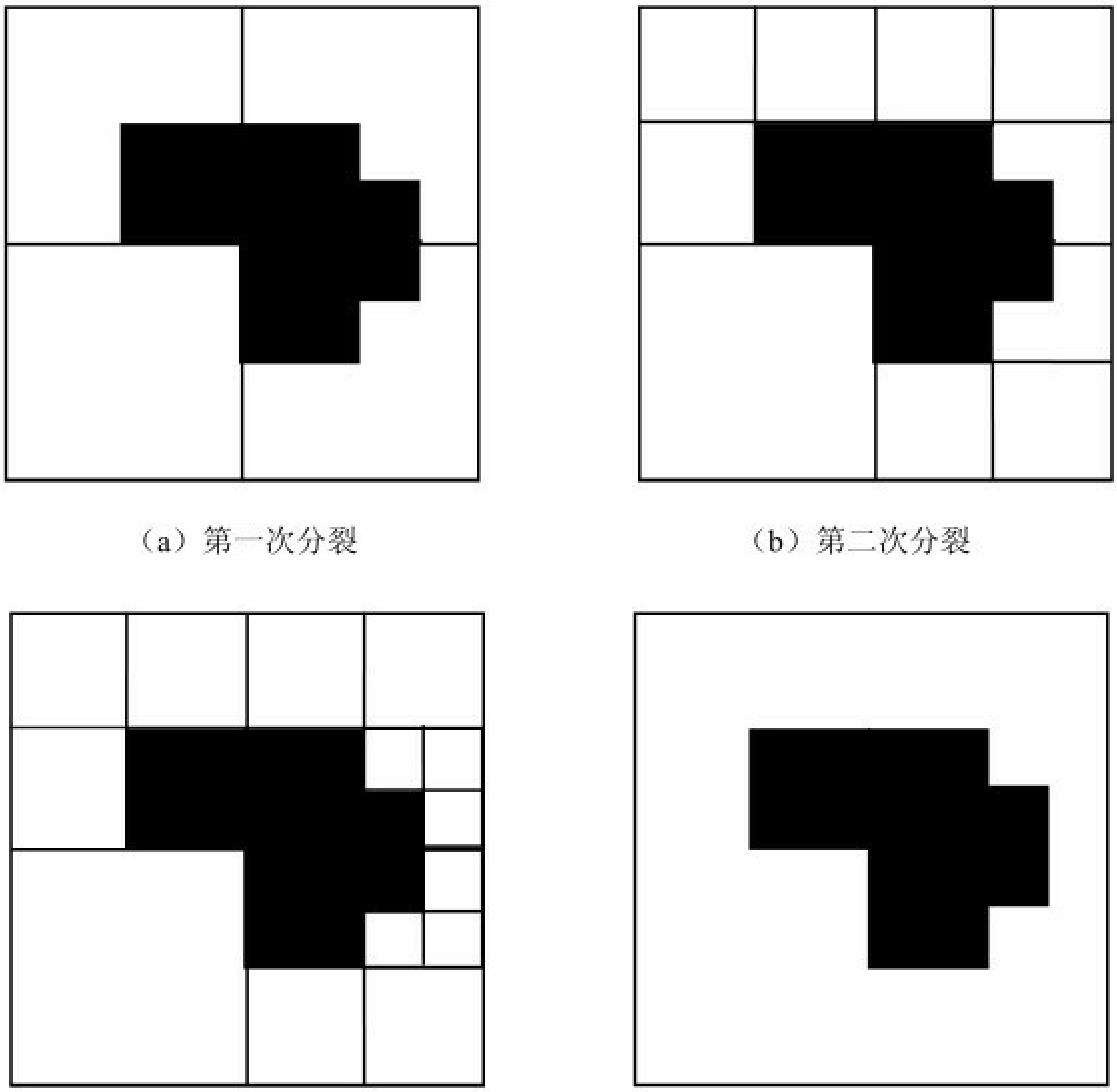
其中图11.20（a）中未标出的4个区域分别为R_{411}、R_{412}、R_{413}和R_{414}。

（2）合并

在分离之后，结果中一般会包含具有满足相似性的邻近区域，这就需要将满足相似性条件的相邻区域进行合并。

可在分裂完成后，也可在分裂的同时，对具有相似特征的相邻区域进行合并。一种方法是将图像中任意两个具有相似特征的相邻区域R_j、R_k合并，即如果$P(R_j \cup R_k)$=TRUE，则合并R_j、R_k。合并的两个区域可以大小不同，即不在同一层。当无法再进行聚合或拆分时操作停止。

一个区域分裂与合并的实例如图11.21所示。



（a）第一次分裂　（b）第二次分裂

（c）第三次分裂　（d）合并

▲图11.21 区域分裂与合并

图像先分裂为如图11.21（a）的区域；第二次分裂时，如图11.21（b）所示，由于左下角区域满足$P(R_i)$=TRUE，则不进行分裂操作；第三次分裂时，如图11.21（c）所示，仅仅右边的突出部分$P(R_i)$=FALSE，需要进行分裂操作，其余不变，完成后，分裂停止；最后，对两个相邻区域实行合并，一直得到最后的结果，如图11.21（d）所示。

区域分裂与合并对分割复杂的场景图像比较有效，如果引入应用领域知识，则可以更好地提高分割效果。

2．区域分裂的**Matlab**实现

在Matlab中，和区域分裂相关的3个主要函数是qtdecomp、qtgetblk和qtsetblk。

（1）qtdecomp函数

Matlab的IPT函数qtdecomp可以进行四叉树分解。该函数首先将图像划分成相等大小的4块，然后对每一个块进行一致性检查：如果该块不符合一致性标准，则将该块继续分为4块，否则不对其进行进一步的分割。这个过程将会一直重复直至每一个块都符合一致性标准，分解的结果可能会包含许多大小不同的块。

qtdecomp函数的常用调用形式为

S=qtdecomp(I ,threshold,[mindim maxdim])

参数说明

● I为输入的灰度图像。

● threshold是分割成的子块中允许的阈值，默认值为0。如果子块中最大元素和最小元素的差值小于该阈值就认为满足一致性条件。对于double型矩阵，threshold将直接作为阈值；而对于uinit8和 uint16类型的矩阵，threshold将被乘以 255和 65 535以作为实际阈值。对于图像而言，threshold的取值范围是0～1。

● [mindim maxdim]是尺度阈值。mindim参数可以屏蔽函数对尺度上小于mindim的子块的处理，而不论这个子块是否满足一致性条件；如果参数形式为[mindim maxdim]，则表示不产生小于mindim尺度的子块，也不保留大于maxdim尺度的子块，此时maxdim/mindim必须是2的整数次幂。

返回值

● S是一个稀疏矩阵，在每个子块的左上角给出子块的大小。

下面的程序给出了对图像I的四叉树分解结果矩阵S。

```
>> I = uint8([1  1  1  1    2  3   6   6;...
              1  1  2  1    4  5   6   8;...
              1  1  1  1    7  7   7   7;...
              1  1  1  1    6  6   5   5;...
              20 22 20 22   1  2   3   4;...
              20 22 22 20   5  4   7   8;...
              20 22 20 20   9  12  40  12;...
              20 22 20 20   13 14  15  16]);
S = qtdecomp(I,.05); %执行4叉树分解，子块阈值为0.05
disp(full(S)); %显示完整的稀疏矩阵
    4    0    0    0    4    0    0    0
    0    0    0    0    0    0    0    0
    0 0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0
    4    0    0    0    2    0    2    0
    0    0    0    0    0    0    0    0
    0    0    0    0    2    0    1    1
    0    0    0    0    0    0    1    1
```

我们看到，S中的非0元素位于该分块的左上角，表示块的大小。

注意

qtdecomp 函数主要适用于边长是 2 的整数次幂的正方形图像，如128×128， 512×512，此时分解可一直进行至子块大小为1×1。

对于长宽不是2的整数次幂的图像，分解可能无法进行到底。例如，对于96×96的图像，将首先分解为48×48，然后是24×24、12×12、6×6，最后是3×3，无法再继续分解。此时必须指定mindim参数为3或是2的整数次幂与3的乘积。

（2）qtgetblk函数

在得到稀疏矩阵S后，利用IPT函数qtgetblk可进一步获得四叉树分解后所有指定大小的子块像素及位置信息。常用调用形式为

[vals, r, c] = qtgetblk(I, S, dim)

参数说明

● I为输入的灰度图像。

● 稀疏矩阵S是I经过qtdecomp函数处理的输出结果。

● dim是指定的子块大小。

返回值

● vals是 dim×dim×k的三维矩阵，包含I中所有符合条件的子块数据。其中 k为符合条件的dim×dim大小的子块的个数，vals(:,:,i)表示符合条件的第 i个子块的内容。

● r和c均为列向量，分别表示图像I中符合条件子块左上角的纵坐标（行索引）和横坐标（列索引）。

下面的程序用于寻找给定图像I中的4×4子块，并返回每个子块的起始点坐标。

```
>> I = [ 1    1    1    1    2    3    6    6
    1    1    2    1    4    5    6    8
    1    1    1    1    10   15   7    7
    1    1    1    1    20   25   7    7
    20   22   20   22   1    2    3    4
    20   22   22   20   5    6    7    8
    20   22   20   20   9    10   11   12
    22   22   20   20   13   14   15   16];
S = qtdecomp(I,5); % 对于double型矩阵I，将以5作为阈值进行四叉树分解
[vals,r,c] = qtgetblk(I,S,4)
```

```
size(vals, 3) %查看4*4子块的个数
ans =
  2
% 显示第1个4*4子块内容
vals(:,:,1) =
   1    1    1    1
   1    1    2    1
   1    1    1    1
   1    1    1    1
% 显示第1个4*4子块内容
vals(:,:,2) =
   20    22    20    22
   20    22    22    20
   20    22    20    20
   22    22    20    20
% 显示子块的位置的左上角坐标
r =
  1
  5
c =
  1
  1
```

（3）qtsetblk函数

在将图像划分为子块后，还需要使用函数qtsetblk将四叉树分解所得到的子块中符合条件的部分全部替换为指定的子块。函数语法为

J = qtsetblk(I, S, dim, vals)

参数说明

● I为输入的灰度图像。

● S是I经过qtdecomp函数处理的结果。

● dim是指定的子块大小。

● vals是 dim×dim×k的三维矩阵，包含了用来替换原有子块的新子块信息。其中 k应为图像I中大小为dim×dim的子块的总数，vals(:,:,i)表示要替换的第 i个子块。

返回值

● J是经过子块替换的新图像。

下面的程序根据三维数组newvals中的内容替换给定图像I中的所有4×4子块。

```
>> I = [ 1    1    1    1    2    3    6    6
    1    1    2    1    4    5    6    8
    1    1    1    1    10   15   7    7
    1    1    1    1    20   25   7    7
    20   22   20   22   1    2    3    4
    20   22   22   20   5    6    7    8
    20   22   20   20   9    10   11   12
    22   22   20   20   13   14   15   16];
S = qtdecomp(I,5); % 对于double型矩阵I，将以5作为阈值进行四叉树分解
newvals = cat(3,zeros(4),ones(4)); %设定欲替换的子块内容
J = qtsetblk(I,S,4,newvals) %根据newvals中的内容替换I中大小为4的子块
J =
   0    0    0    0    2    3    6    6
   0    0    0    0    4    5    6    8
   0    0    0    0    10   15   7    7
```

0　0　0　0　20　25　7　7

1　1　1　1　1　2　3　4

1　1　1　1　5　6　7　8

1　1　1　1　9　10　11　12

1　1　1　1　13　14　15　16

我们看到，I中左半部分的2个4×4子块分别被替换为0子块和1子块（J的左半部分）。

注意

sizeof(newvals,3)必须和I中指定大小的子块数目相同，否则qtsetblk会提示错误信息。

一个区域分割的综合实例如例11.3所示。

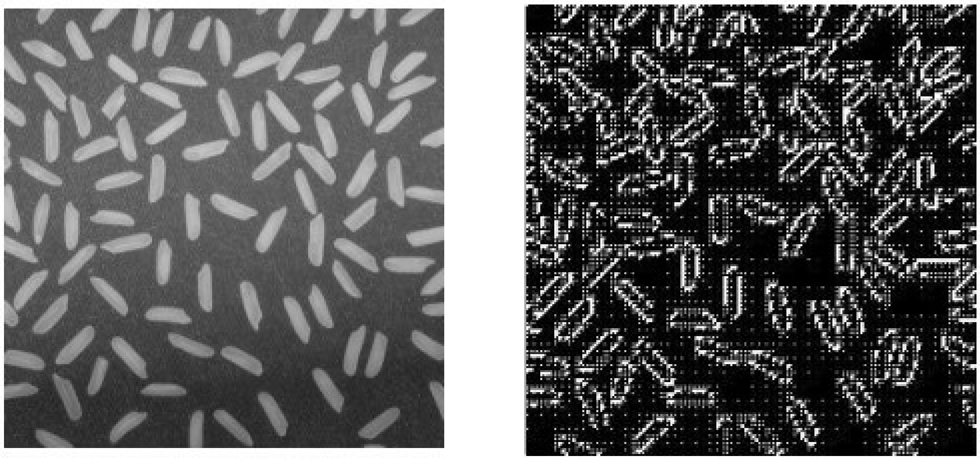
[例11.3]对Matlab示例图片rice.png进行四叉树分解，并以图像形式显示所得的稀疏矩阵。同时，取得所有子块和符合各种维度条件的子块数目。

相应的Matlab实现代码如下。

```
I1 = imread('rice.png'); %读入原图像
imshow(I1) %得到图11.22（a）
%选取阈值为0.2，对原始图像进行四叉树分解
S = qtdecomp(I1,0.2);
%原始的稀疏矩阵转换为普通矩阵，使用full函数
S2 = full(S);
figure;
imshow(S2); %得到图11.22（b）
ct = zeros(6, 1); %记录子块数目的列向量
% 分别获得不同大小块的信息，子块内容保存在三维数组vals1～val6中，子块数目保存在ct向量中
for ii = 1:6
```

```
    [vals{ii},r,c] = qtgetblk(I1,S2,2^(ii-1));
    ct(ii) = size(vals{ii},3);
end
```

上述程序的运行结果如图11.22所示。



（a）原始图像　　（b）四叉树分解结果

▲图11.22 四叉树分解算法

11.6 基于形态学分水岭算法的图像分割

在第10章，我们曾学习了形态学。实际上，形态学也可以应用到图像分割上，而且通常它生成的分割结果更稳定。本节主要介绍基于形态学分水岭算法的图像分割。

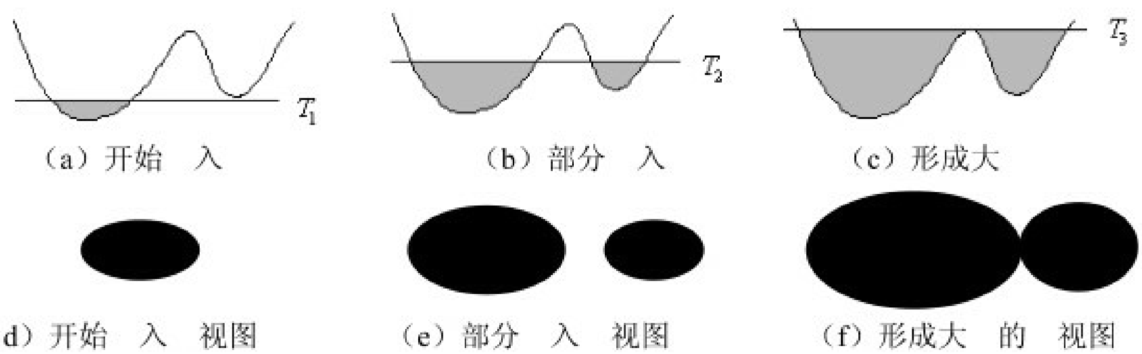
11.6.1 形态学分水岭算法

所谓分水岭算法（watershed），是一种基于拓扑理论的形态学分割方法，其基本思想是把图像看作是测地学上的拓扑地貌，图像中每一点像素的灰度值表示该点的海拔高度，高灰度值对应着山峰，低灰度值处对应着山谷。每一个局部极小值及其影响区域称为集水盆，而集水盆的

边界则形成分水岭。

1．算法原理

分水岭的概念和形成可以通过模拟浸入过程来说明。在每一个局部极小值表面，刺穿一个小孔，然后把整个模型慢慢浸入水中，随着浸入的加深，每一个局部极小值的影响域慢慢向外扩展，在两个集水盆汇合处构筑大坝，即形成分水岭，这个分水岭就是我们要求的边界。这一过程如图 11.3所示。



▲图11.23 分水岭算法示意图

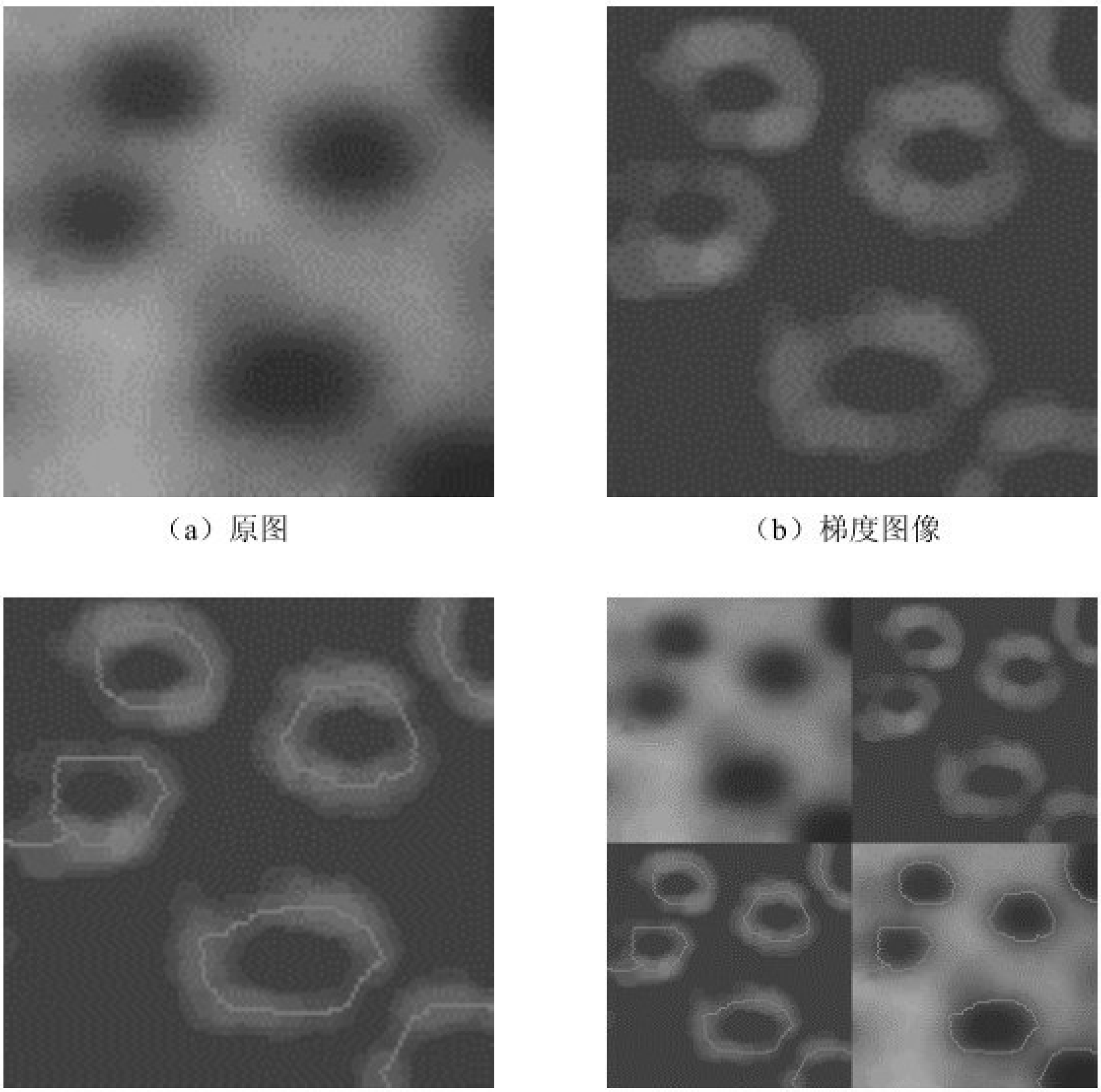
图11.23所示是模型浸入水中灰度值的截面图，说明了模型浸入水中慢慢形成分水岭的过程。图11.23（a）是刚刚浸入水中时，水从模型的最低洼处涌入，在第一个集水盆处慢慢上升，其俯视图为图11.23（d）；图11.23（b）是当模型浸入到一定程度时，水分别在第一个和第二个集水盆中上升，其俯视图如图11.23（e）所示；图11.23（c）是当第一、二个集水盆的水将连未连时的截面图，其俯视图为图11.23（f），此时两个集水盆的边界就是大坝，大坝的边界就是所要求的分割线，即原始图像的边缘。

2．实现方法

用形态学来实现分水岭分割算法时，先利用腐蚀将每个粒子腐蚀成一个点而不消失，再利用条件膨胀标记出生长出的粒子（条件膨胀就是

将膨胀限制在原始集合内，同时当两个粒子靠得很近时不能有连接部分）。

分水岭分割方法应用在梯度图像上,那么集水处在理论上就对应灰度变化最小的区域,而分水岭就对应灰度变化相对最大的区域。如图11.24所示，其中图（d）是分水岭叠加在原图上的图像。

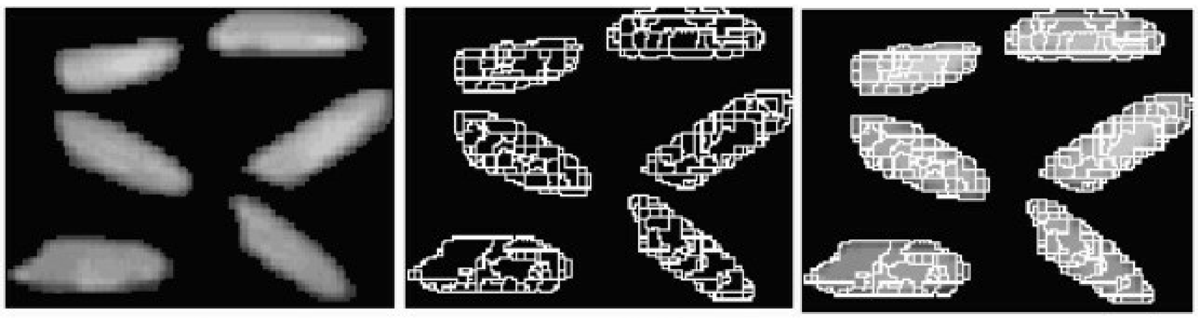


（a）原图　（b）梯度图像　（c）梯度图的分水岭　（d）分割结果

▲图11.24 分水岭算法应用

3．过渡分割问题

分水岭算法对微弱边缘具有良好的响应，能够得到封闭连续的边缘，这为分析图像的区域特征提供了很大的便利，也是其同其他分割算法相比的一个主要优势。然而，分水岭算法也有一个致命的弱点，对于边缘的强响应使其对噪声和物体表面细微的灰度变化非常敏感，从而导致常常产生过度分割结果，如图11.25所示。



（a）原图　　（b）分水岭　　（c）分割结果

▲图11.25 分水岭算法的过度分割

从图11.25中可见，原本完整的物体却被分割成了很多更细小的区域，这就是过度分割。显然，这是我们所不希望的结果，因此必须采用某些方法来改进分水岭算法。

针对这个问题，有很多种改进的分水岭分割技术。综合来讲，主要有以下3类。

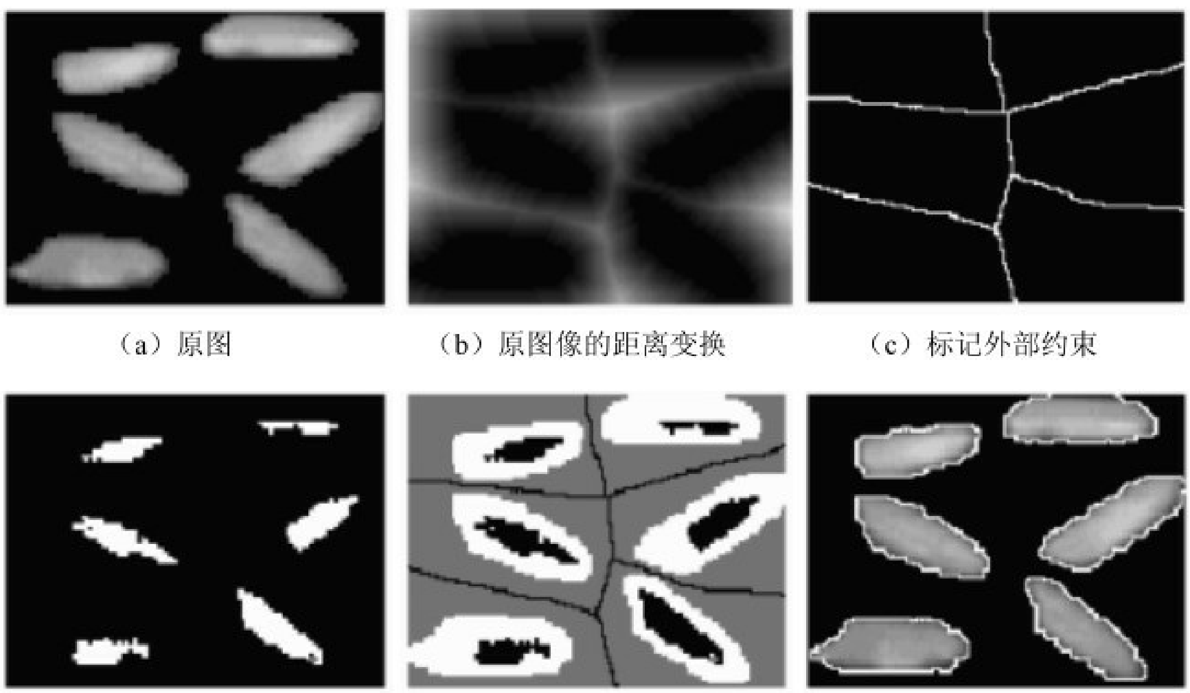
（1）在应用分水岭分割之前对图像进行一些预处理，诸如除噪、求梯度图像、形态学重构、标记前景背景等，这些操作的目的是减少小的积水盆，从而减少过分割区域的数量。

（2）分割时添加约束，通过约束条件来限制分割过程。

（3）在应用分水岭分割之后对结果图像进行合并处理。如果初始分割产生过多小区域，合并处理会具有很大的运算量，所以后处理的时间复杂度经常较高。此外，合并准则的确定也是一件比较麻烦的事情，

通常有基于相邻区域的平均灰度信息和边界强度信息的合并准则，不同的合并准则会得到不同的分割结果。

图11.26所示为准确标记的分水岭分割过程。这是通过合并预处理步骤来限制允许存在的区域的数目。一个标记是属于一幅图像的联通分量。选择标记时，既要有与重要对象相联系的内部标记，也要有与背景相联系的外部标记。一般来说，首先预处理图像，然后定义一个所有标记必须满足的准则，即定义内标记和外标记。



（a）原图　（b）原图像的距离变换　（c）标记外部约束

（d）标记内部约束　（e）由标记内外部约束重构的梯度图　（f）分割结果

▲图11.26 准确分割的分水岭算法过程

将图11.26中的图（f）与图11.25中的图（c）相比较，显然，采用准确标记的分水岭分割后，图像中物体的边界得到准确的检测，其改善之处是显而易见的。

11.6.2 Matlab实现

Matlab中提供了原始的分水岭函数watershed，这个函数提供了基本的分水岭算法处理能力——但也仅限于此。因此，您需要事先对图像进行必要的处理，否则，这个函数返回的结果将会是很显然的过度分割结果。我们首先来介绍一下这个函数。

watershed函数使用Fernand Meyer算法对输入的图像进行分水岭算法的处理，它支持数值矩阵和逻辑矩阵，但不支持稀疏格式存储的矩阵。我们首先介绍这个函数的语法：

L = watershed(A)

L = watershed(A, conn)

参数说明

●A指定需要寻找轮廓的原始图像。

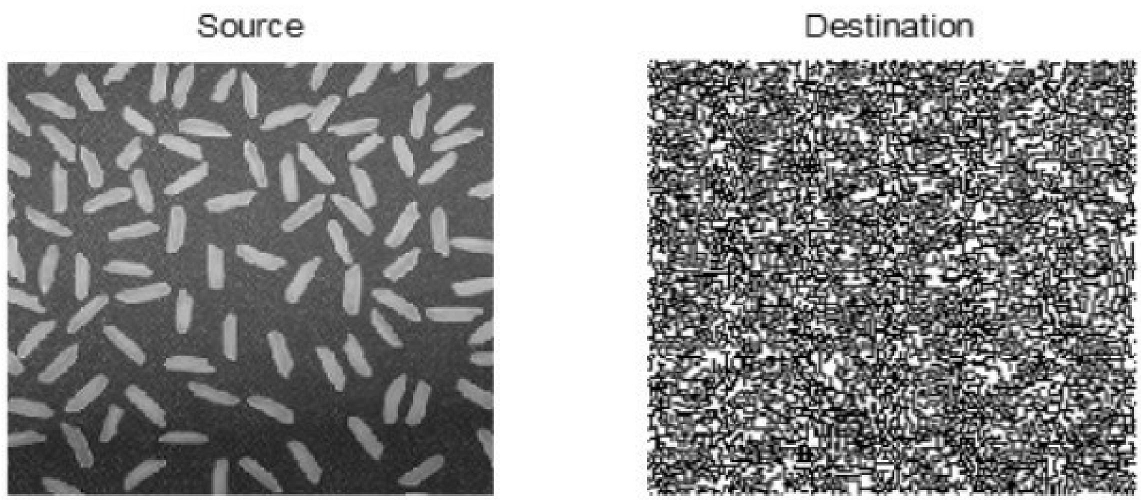
● 可选参数conn指定分水岭算法将要考虑的邻域数量，对于二维图像，这个值可以为4或者8；而这个函数也可以处理三维图像，因此这里可以有不同的取值。

返回值

● 输出的矩阵A中数值为0的元素表明不属于任何一个划分出的区域，而各个分水岭分割出的区域被用不同的序号表示。输出的矩阵是double 类型，因此，如果要做一个简单的示例，只需要直接将输出的矩阵使用imshow函数显示即可。因为，数值为0的元素显示为黑色，而其他的元素（属于某个分区的元素）的值一定会大于等于1，因此会显示为白色。

因此，我们首先观察下面例子及其输出图11.27。

```
>> i = imread('rice.png');
>> d = watershed(i);
>> subplot(1,2,1); imshow(i); title('Source');
>> subplot(1,2,2); imshow(d); title('Destination');
```



▲图11.27 Matlab分水岭算法示例

很显然，我们得到了一个严重过度分割的分水岭算法结果。事实上，我们可以对输入图像进行必要的前期处理以避免这种结果。Matlab没有直接将这些前期算法内置到watershed函数中，因此需要我们手工进行这些操作。一种前期处理方法是将图像转换为二值图像，再进行距离变换。

在本章后面的 Matlab 综合案例一节，我们将使用同样的一个例子进行分水岭算法的处理，请读者可从中体会各种前期处理方法的效果差别。

11.7 Matlab综合案例——分水岭算法

前文提到的单纯的分水岭算法存在着严重的过度分割问题，但这个算法实质上是一种很有效的图像分割方法。事实上，只要对其添加必要的前期处理步骤，就可以大大改善处理的效果。下面我们讲解几种常用的前期处理算法，请注意，算法中的参数对于不同的图像有不同的最优值，请读者自行探索调试。

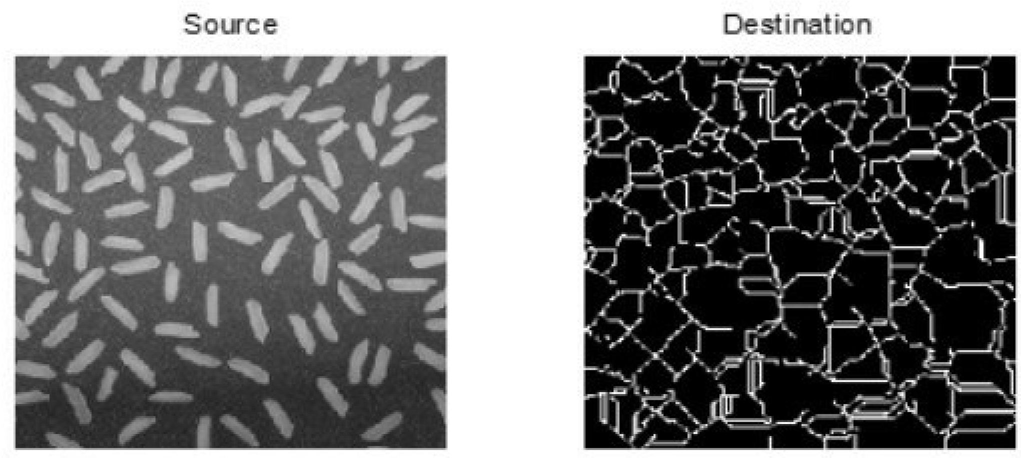
1．前期处理方法：二值图像距离变换

这种方法将源图像首先按照阈值法转换为二值图像，然后对二值图像进行距离变换。Matlab中执行二值图像距离变换的函数是bwdist。距离变换的定义很简单，就是将图像中的每个像素的值变换为这个像素到离它距离最近的非零像素的距离。这里的距离是按照指定的距离计算方式进行的，通常为8-邻域概念下的距离度量。很显然，在这种定义下，非零像素的距离变换值将为0，而零像素的距离变换值将为非零值。我们不再单独详细介绍 bwdist 函数的用法，请读者从下面的例子中自行体会。如果理解存在困难，可以使用help bwdist或者doc bwdist命令获取在线帮助。

在下面的例子中，我们还是使用相同的一幅示例图片，观察使用二值化图像距离变换进行源图像的前期处理之后，分水岭算法的处理效果。例子的代码如下。

```
i = imread('rice.png');
b = im2bw(i, graythresh(i));
bc = -b;
d = bwdist(bc);
l = watershed(-d);
w = l == 0;
subplot(1,2,1); imshow(i); title('Source');
subplot(1,2,2); imshow(w); title('Destination');
```

这个例子的运行结果如图11.28所示，读者可以将这张图片与前面的过度分割结果对比，这里的差距是非常明显的。



▲图11.28 改善的分水岭分割结果

2．前期处理方法：梯度处理

这种前期处理方法是使用梯度算子处理原始的灰度图像。对灰度图像而言，在图像中物体边缘的部分，梯度值是相对更大的，而在其他部分的梯度值都相对小得多。因此，对使用梯度算法处理后的图像执行分水岭算法处理得到的结果将是分水岭线存在于图像的边缘上，从而将图像的区域分开。下面提供一个例子说明这种前期处理方法。

这个例子使用前文提到的Sobel梯度算子对图像进行梯度操作。使用特定算子对图像进行滤镜操作的函数是fspecial，其使用语法为

h = fspecial(type, parameters)

其中parameters为可选参数，type为滤镜名称，如果要使用索贝尔算子进行梯度滤镜操作，则type的值应为'sobel'。fspecial函数返回一个可以用于imfilter函数的滤镜。imfilter函数用于对N维图像进行滤镜操作，其语法为

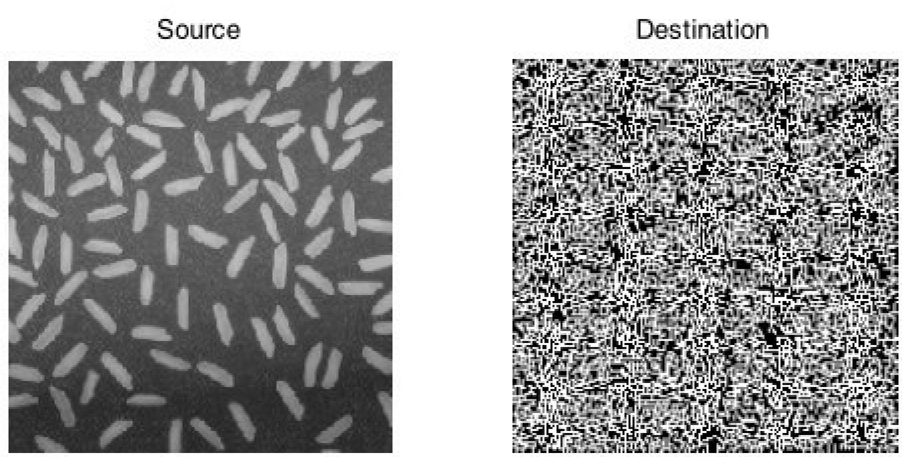
B = imfilter(A,H,options…)

其中H为使用fspecial函数指定的滤镜。在这个例子中，我们要使用imfilter对图像进行索贝尔梯度滤波。于是，这个例子的算法实现如下。

i = imread('rice.png');

```
h = fspecial('sobel');
id = im2double(i);
g = sqrt(imfilter(id, h, 'replicate') .^ 2 + imfilter(id, h, 'replicate') .^ 2);
l = watershed(g);
w = l == 0;
subplot(1,2,1); imshow(i); title('Source');
subplot(1,2,2); imshow(w); title('Destination');
```

本例的处理结果如图11.29所示，可见，梯度算法并不一定比二值图像方法更好，在这个例子中就出现了明显的过度分割。我们可以尝试改进一下这种梯度处理方法，在进行梯度算子处理之前应用图像平滑处理，这样就可以尽量减小梯度算子对图像噪声的放大（正如微分运算对高频噪声的放大一样）。



▲图11.29 梯度算法预处理的分水岭算法结果

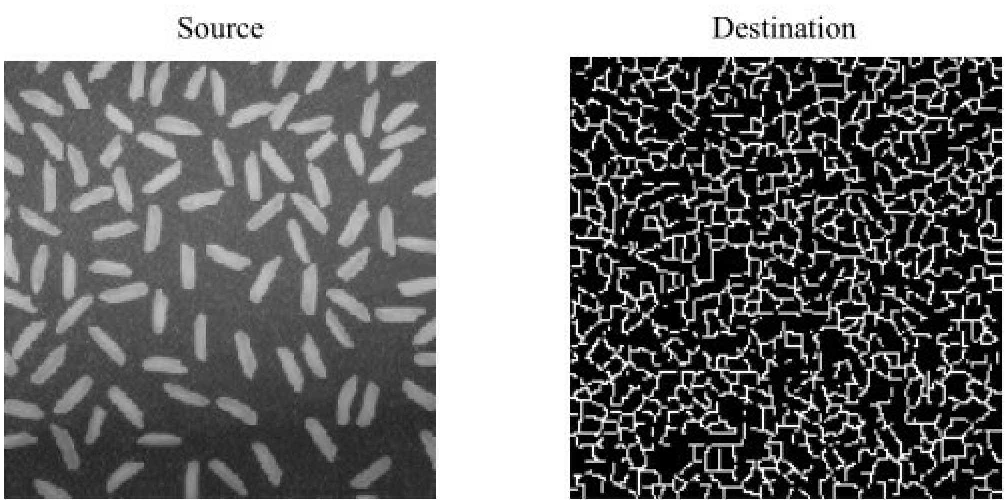
为了达到平滑图像的目的，我们可以在加载图像和应用梯度滤镜之后，对图像进行如下操作。

```
sm = imclose(imopen(g, ones(3,3)), ones(3,3));
```

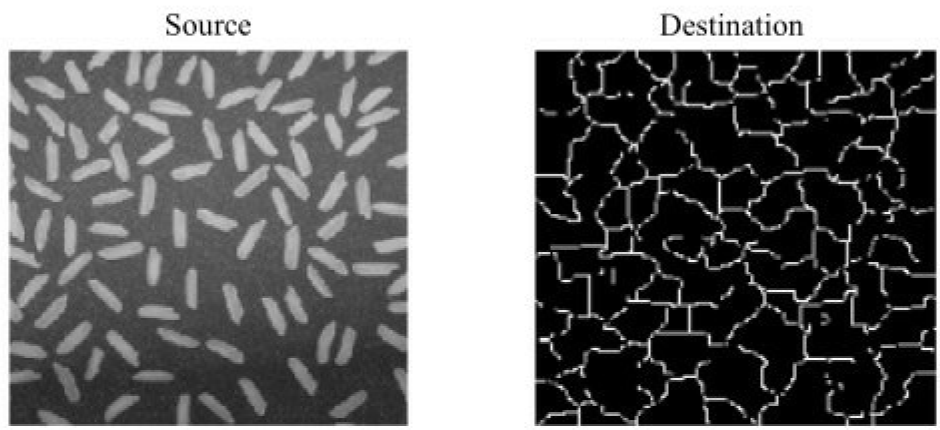
这样，上面的算法就变为

```
i = imread('rice.png');
h = fspecial('sobel');
id = im2double(i);
g = sqrt(imfilter(id, h, 'replicate') .^ 2 + imfilter(id, h, 'replicate') .^ 2);
sm = imclose(imopen(g, ones(3,3)), ones(3,3));
l = watershed(sm);
w = l == 0;
subplot(1,2,1); imshow(i); title('Source');
subplot(1,2,2); imshow(w); title('Destination');
```

本例的处理结果如图11.30所示，可以发现，过度分割的情况好了一些，但是还存在一定的问题。可以通过调整图像平滑算法中的矩阵大小来改善这种情况。事实上，图11.31就是那两个矩阵的大小是7×7的情况，可以看到过度分割的情况有很大改善。



▲图11.30 使用 5×5大小模板平滑后的结果



▲图11.31 使用 7×7模板平滑后的结果

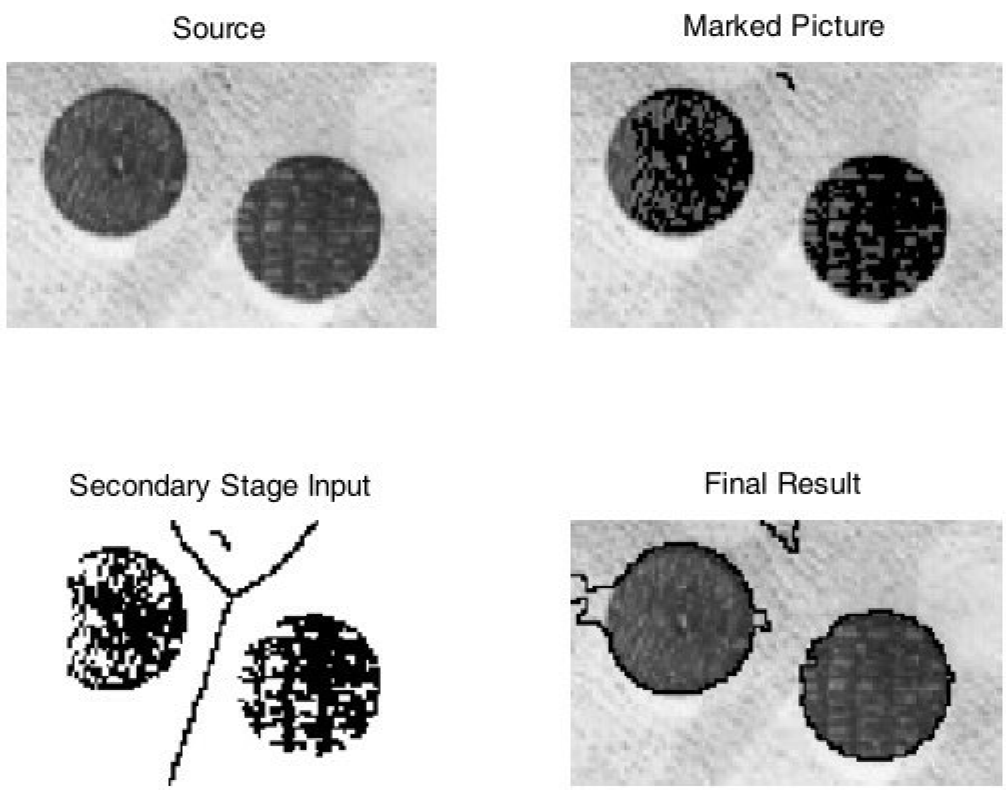
3．前期处理方法：标记约束

前文提到的二值图像处理和梯度算子处理方法都存在一个共同的缺点，就是难以对图像中的噪声产生足够的抵抗力，因而很容易造成图像分割的混乱。另外，这两种方法对过度分割的预防均不甚完善。

为此，我们采用了标记约束的分水岭算法。很完善的标记算法会使程序较为复杂，因此我们采用一个较为简单的示例图像来完成这个例子。需要注意的是，本例中所提供的程序对于处理背景仅有噪声而没有明显渐变的图片效果较好，同时，最佳状态需要是背景颜色相对前景颜色更浅。这个例子的处理结果如图11.32所示。

```
f = imread('test.tif');
subplot(2,2,1);imshow(f);title('Source');
h = fspecial('sobel');
fd = double(f);
g = sqrt(imfilter(fd, h, 'replicate') .^ 2 + imfilter(fd, h, 'replicate') .^ 2);
im = imextendedmin(f,50);
fim = f;
fim(im) = 0;
```

```
subplot(2,2,2);imshow(fim);title('Marked Picture');
lim = watershed(bwdist(im));
em = lim == 0;
g2 = imimposemin(g, im | em);
subplot(2,2,3);imshow(g2);title('Secondary Stage Input');
l2 = watershed(g2);
f2 = f;
f2(l2==0) = 0;
subplot(2,2,4);imshow(f2);title('Final Result');
```



▲图11.32 使用标记约束方法的改善情况

下面再给出一个对rice图片进行分水岭变换且效果相对较好的例子。例11.4给出了Matlab的代码实现。

[例11.4]基于分水岭变换的米粒图像分割。

afm=imread('rice.png');

figure,imshow(afm); % 图11.33（a）

se = strel ('disk', 15) ;

Itop = imtophat (afm , se) ;

figure , imshow(Itop , [ ]) , title ('top - hat image'); % 图11.33（b）

Ibot = imbothat (afm , se) ;

figure , imshow(Ibot , [ ]) , title ('bottom - hat image'); % 图11.33（c）

Ienhance = imsubtract (imadd (Itop , afm) , Ibot) ;

figure , imshow(Ienhance) , title ('original + top - hat - bottom - hat');% 图11.33（d）

Iec = imcomplement (Ienhance) ;

figure , imshow(Iec) , title ('complement of enhanced image'); % 图 11.33（e）

Iemin = imextendedmin(Iec , 22) ;

figure , imshow(Iemin) , title ('extended minima image’'); % 图 11.33（f）

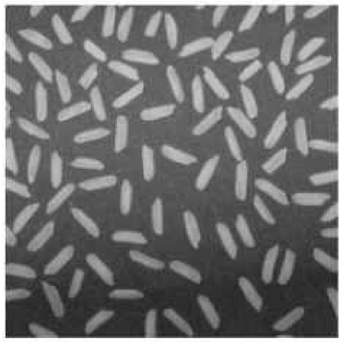
Iimpose = imimposemin (Iec , Iemin) ;

figure , imshow(Iimpose) , title ('imposed minima image'); % 图 11.33（g）

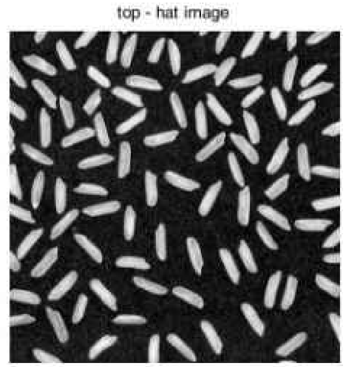
BW = watershed(Iimpose);

Imshow(BW); % 图11.33（g）

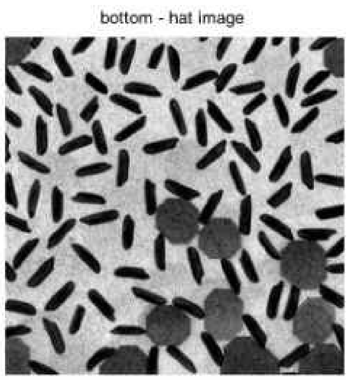
分割结果如图11.33所示。



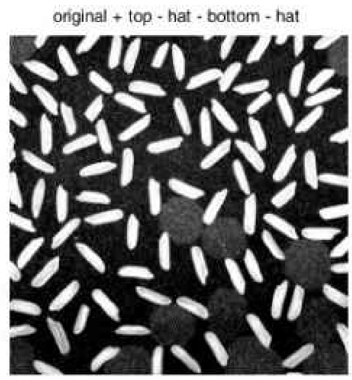
（a）原始图像



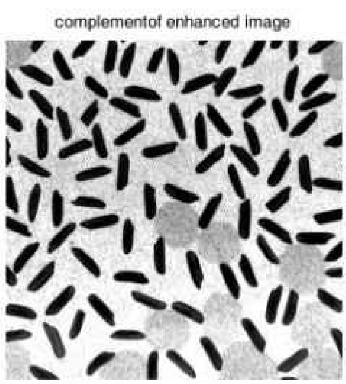
（b）顶帽变换



（c）底帽变换



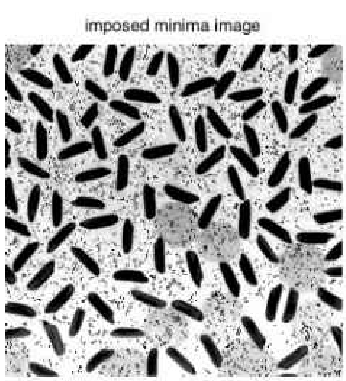
（d）补偿后的图像



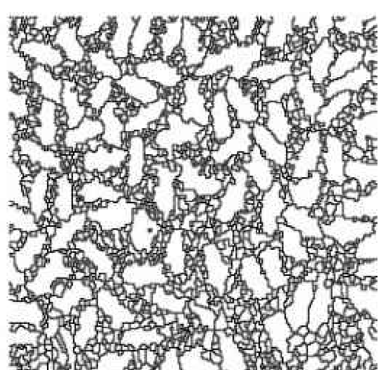
（e）完整增强后的图像



（f）最小化标记约束的图像



（g）将（f）反相的结果



（h）最终结果

▲图11.33 改进的分水岭算法分割结果

11.8 小结

图像分割问题是一个十分困难的问题。因为分割后的图像是系统目标的一个函数，所以根本不存在理想的或正确的分割。

物体及其组成部件的二维表现形式受到光照条件、透视畸变、观察点变化、遮挡等的影响。此外，物体及其组成部件与背景之间在视觉上可能是无法区分的。因此，人们无法预测能够从图像中抽取出哪些与物体识别相关的初始信息。

唯一可以肯定的事情是，这一过程将在本质上具有不可靠性。某些有用的信息能够被抽取出，但同时也会出现许多错误。因此，在任何应用领域中都不存在最优解。分割结果的好坏或者正确与否，目前还没有一个统一的评价判断标准，大都从分割的视觉效果和实际的应用场景来判断。

第12章 特征提取

从本章开始，我们将逐步从数字图像处理向图像识别过渡。严格地说，图像特征提取属于图像分析的范畴，是数字图像处理的高级阶段，同时也是图像识别的开始。

本章的知识和技术热点

● 常用的基本统计特征，如周长、面积、均值等区域描绘子，以及直方图和灰度共现矩阵等纹理描绘子。

● 主成份分析（PCA, Principal Component Analysis）。

● 局部二进制模式（LBP, Local Binary Pattern）。

本章的典型案例分析

● 基于PCA技术的人脸数据集的降维处理。

12.1 图像特征概述

众所周知，计算机是不认识图像，只认识数字的。为了使计算机能够“理解”图像，从而具有真正意义上的“视觉”，本章中我们研究如何从图像中提取有用的数据或信息，得到图像的“非图像”的表示或描述，如数值、向量和符号等。这一过程就是特征提取，而提取出来的这些“非图像”的表示或描述就是特征。有了这些数值或向量形式的特征我们就可以通过训练过程教会计算机如何懂得这些特征，从而使计算机具有了识别图像的本领。

12.1.1 什么是图像特征

特征是某一类对象区别于其他类对象的相应（本质）特点或特性，或这些特点和特性的集合。特征是通过测量或处理能够抽取的数据。对

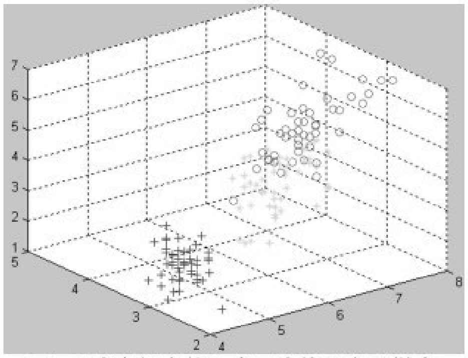
于图像而言，每一幅图像都具有能够区别于其他类图像的自身特征，有些是可以直观地感受到的自然特征，如亮度、边缘、纹理和色彩等；有些则是需要通过变换或处理才能得到的，如矩、直方图以及主成分等。

12.1.2 图像特征的分类

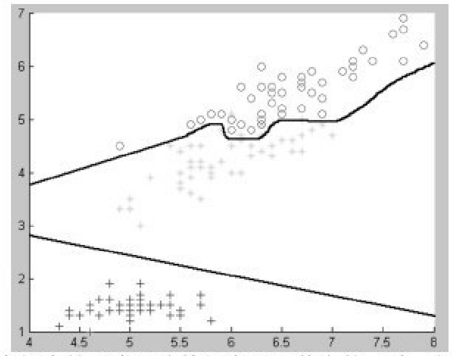
图像特征的分类有多种标准。如从特征自身的特点上可以将其分为两大类：描述物体外形的形状特征，以及描述物体表面灰度变化的纹理特征。而从特征提取所采用的方法上我们又可以将特征分为统计特征和结构（句法）特征。由于本书后面的图像识别部分主要着眼于统计模式识别，不涉及句法模式识别的具体内容。因此本章中也主要讨论图像的统计特征及其获取方法。

12.1.3 特征向量及其几何解释

我们常常将某一类对象的多个或多种特性组合在一起，形成一个特征向量来代表该类对象，如果只有单个数值特征，则特征向量为一个一维向量；如果是n个特性的组合，则为一个n维特征向量。该特征向量常常被作为识别系统的输入。实际上，一个n维特征就是一个位于n维空间中的点，而识别（分类）的任务就是找到对这个n维空间的一种划分，如图12.1所示。在后面各章的讨论中，一般将待分类的对象称为样本，将其特征向量称为样本特征向量或样本向量。



（a）三维空间中的 3 类三维特征向量样本



（b）二维空间中的 3 类二维特征向量及其上的一种可能的划分

▲图12.1 特征向量的几何解释

比如说我们要区分3种不同的鸢尾属植物，可以选择其花瓣长度和花瓣宽度作为特征，这样就以1个三维特征代表1个植物对象，比如（5.1, 3.5）。如果再加上萼片长度和萼片宽度，则每个鸢尾属植物对象由一个四维特征向量表示，如（5.1, 3.5, 1.4, 0.2）。

12.1.4 特征提取的一般原则

图像识别实际上是一个分类的过程，为了识别出某图像所属的类别，我们需要将它与其他不同类别的图像区分开来。这就要求选取的特征不仅要能够很好地描述图像，更重要的是还要能够很好地区分不同类别的图像。

我们希望选择那些在同类图像之间差异较小（较小的类内距），在不同类别的图像之间差异较大（较大的类间距）的图像特征。我们称之为最具有区分能力（most discriminative）的特征。此外，在特征提取中先验知识扮演着重要的角色，如何依靠先验知识来帮助我们选择特征也是后面将持续关注的问题。

要对某个图像进行分类，应如何提取该图像的特征呢？一个最容易想到的方法是提取图像中所有像素的灰度值作为特征，这样可以提供尽可能多的信息给识别程序（分类器），让分类器具有最大的工作自由度

（基于神经网络的很多特征提取就是这样做的，见第14章）。然而，高维度意味着高计算复杂度，这为后续的处理和识别带来了巨大的困难（见12.3.1小节维度灾难）。此外，很多时候由于我们已经掌握了有关样本图像的某些先验知识，使得这种把全部像素信息都交给分类器的做法显得没有必要。如我们已经知道鼻子、肤色、面部轮廓等信息与表情关联不大，那么在表情识别中就不需要人脸照片中的全部信息。可以只拿出眉毛、眼睛和嘴这些表情区域作为特征提取的候选区，这时可以进一步在表情区中提取统计特征。

12.1.5 特征的评价标准

一般来说，特征提取应具体问题具体分析，其评价标准具有一定的主观性。然而，还是有一些可供遵循的普遍原则，能够作为我们在特征提取实践中的指导。总结如下。

- 特征应当容易提取。换句话说，为了得到这些特征我们付出的代价不能太大。当然，这还要与特征的分类能力权衡考虑。
- 选取的特征应对噪声和不相关转换不敏感。例如要识别车牌号码，车牌照片可能是从各个角度拍摄的，而我们关心的是车牌上字母和数字的内容，因此就需要得到对几何失真变形等转换不敏感的描绘子，从而得到旋转不变，或是投影失真不变的特征。
- 最重要的一点，总是应试图寻找最具区分能力的特征。

12.2 基本统计特征

本节将主要介绍一些常用的基本统计特征，包括一些简单的区域描绘子、直方图及其统计特征，以及灰度共现矩阵等。这些特征具有简单、实用的优势，同时首先学习它们将有助于读者理解统计特征的特

点，从而更好地把握本章后面的内容。

12.2.1 简单的区域描绘子及其Matlab实现

在经过图像分割得到各种我们感兴趣的区域之后，可以利用下面将要介绍的一些简单区域描绘子作为代表该区域的特征。通常将这些区域特征组合成特征向量以供分类使用。

常用的简单区域描绘子如下。

- 周长：区域边界的长度，即位于区域边界上的像素数目；
- 面积：区域中的像素总数；
- 致密性：（周长）^{2}/面积；
- 区域的质心；
- 灰度均值：区域中所有像素的平均值；
- 灰度中值：区域中所有像素的排序中值；
- 包含区域的最小矩形；
- 最小或最大灰度级；
- 大于或小于均值的像素数；
- 欧拉数：区域中的对象数减去这些对象的孔洞数。

在Matlab中，函数regionprops是用于计算区域描绘子的有利工具，其原型为

D=regionprops(L, properties);

参数说明

- L是一个标记矩阵，可通过10.3.4小节介绍的连通区标注函数bwlabel得到。
- properties可以是一个用逗号分割的字符串列表，其一些常用取值如表12.1所示。

表12.1

合 法 值	含 义
'Area'	区域内的像素总数
'BoundingBox'	包含区域的最小矩形。1×4 向量：[矩形左上角 x 坐标，矩形左上角 y 坐标，x 方向长度，y 方向长度]
'Centroid'	区域的质心。1×2 向量：[质心 x 坐标，质心 y 坐标]
'ConvexHull'	包含区域的最小凸多边形。P×2 矩阵：每一行包含多边形 p 个顶点之一的 x 和 y 坐标
'EquivDiameter'	和区域有着相同面积的圆的直径
'EulerNumber'	区域中的对象数减去这些对象的孔洞数

返回值

● D是一个长度为max(L(:))的结构数组，该结构的域表示每个区域的不同度量，具体取决于properties指定要提取的度量类型。

[例12.1]利用regionprops函数提取简单的区域特征。

如要提取二值图像bw_mouth.bmp中每个区域的面积和质心作为特征，相应代码如下。

```
>> I = imread('../bw_mouth.bmp'); %读入二值图像，文件
bw_mouth.bmp位于在线资源第12章的根目录
中
>> Il = bwlabel(I); %标注连通区，得到标记矩阵Il
>> D = regionprops(Il, 'area', 'centroid'); %提取面积和质心
>> D % 查看返回的结构体
D =
4x1 struct array with fields:
  Area
  Centroid
>> D.Area % 4个连通区域的面积
ans =
  92
ans =
  713
```

```
ans =
  1
ans =
  1
>> v1 = [D.Area] % 将面积转存为向量
v1 =
  9    2    713    11
>> D.Centroid % 4个连通区域的质心
ans =
  5.1304    31.4783
ans =
  29.8597    21.6227
ans =
  10    36
ans =
  63    36
>> v2 = [D.Centroid] %将质心转存为向量
v2 =
  5.1304 31.4783 29.8597 21.6227 10.0000 36.0000 63.0000 36.0000
```

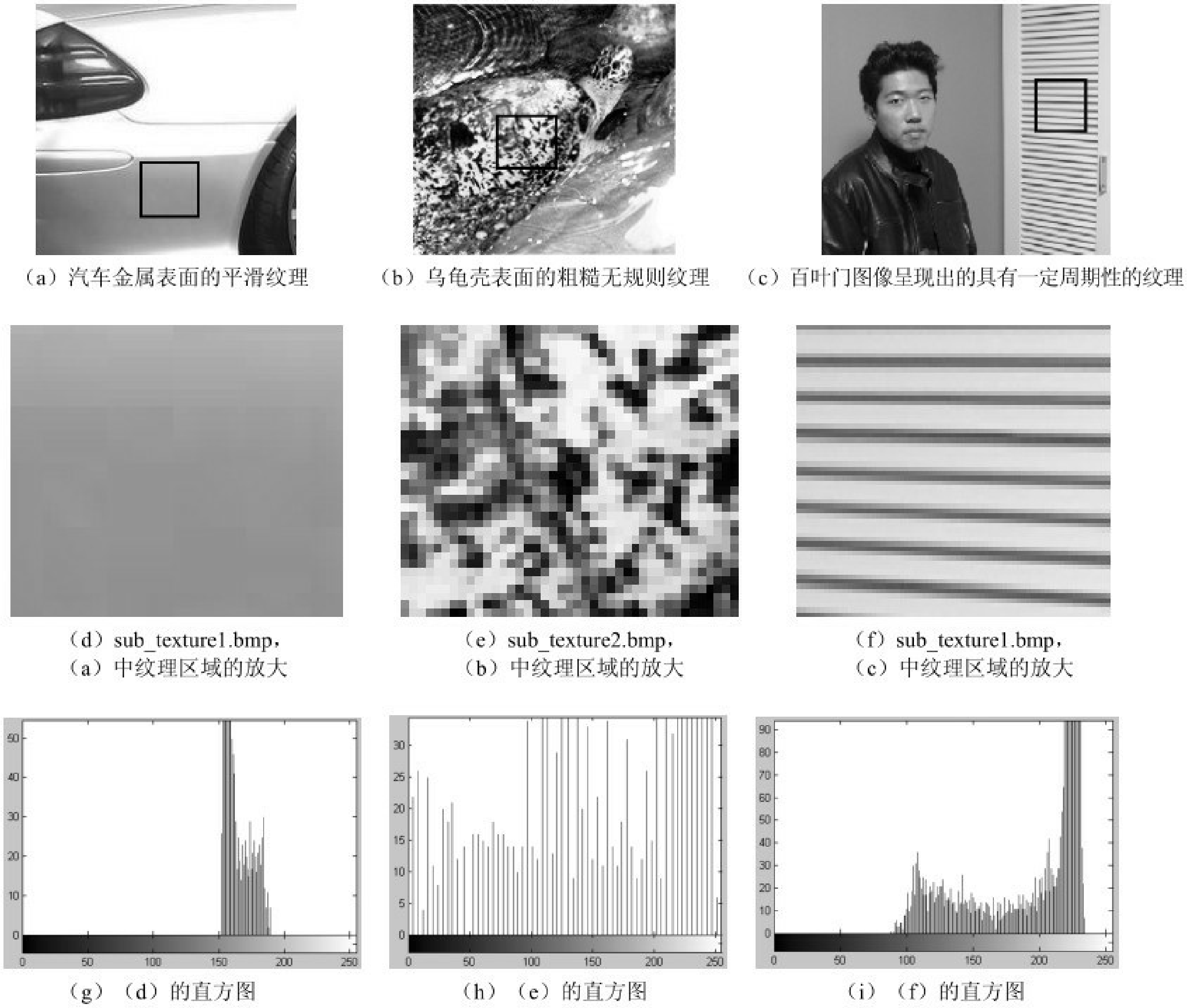
12.2.2 直方图及其统计特征

在第3章点运算中，我们已经学习过直方图的概念和计算方法，当时直方图更多的是作为一种辅助图像分析的工具。这里则要将直方图作为图像纹理描述的一种有力手段，以直方图及其统计特征作为描述图像的代表性特征。

首先来看纹理的概念。纹理是图像固有的特征之一，是灰度（对彩

色图像而言是颜色）在空间以一定的形式变换而产生的图案（模式），有时具有一定的周期性。图12.2（d）～（f）所示为3种不同特点的纹理：金属表面的平滑纹理、龟壳表面的粗糙无规则纹理，以及百叶门图像中具有一定周期性的纹理。既然纹理区域的像素灰度级分布具有一定的形式，而直方图正是描述图像中像素灰度级分布的有力工具，因此用直方图来描述纹理就顺理成章了。

毫无疑问，相似的纹理具有相似的直方图；图12.2所示为3种不同特点的纹理对应着3种不同的直方图。这说明直方图与纹理之间存在着一定的对应关系。因此，我们可以用直方图或其统计特征作为图像纹理特征。直方图本身就是一个向量，向量的维数是直方图统计的灰度级数，因此我们可以直接以此向量作为代表图像纹理的样本特征向量，从而交给分类器处理，对于LBP直方图就常常这样处理（见12.5节）；另一种思路是进一步从直方图中提取出能够很好地描述直方图的统计特征，将直方图的这些统计特征组合成为样本特征向量，这样做的好处是大大降低了特征向量的维数。



▲图12.2 三种不同特点的纹理

直方图的常用统计特征包括以下内容。

（1）均值：纹理平均亮度的度量。

$$m=\sum_{i=0}^{L-1} z_i p(z_i) \quad 或 \quad m=\frac{\sum_{i=0}^{L-1} z_i h(z_i)}{\sum_{i=0}^{L-1} h(z_i)} \tag{12-1}$$

式中：L是灰度级总数；z_i表示第i个灰度级；$P(z_i)$是归一化直方图

灰度级分布中灰度为z_i的概率；$h(z_i)$表示直方图中统计的灰度为z_i的像素个数（不需要归一化）。

（2）标准方差：纹理平均对比度的度量。

$$\sigma=\sqrt{\sum_{i=0}^{L-1}(z_i-m)^2 p(z_i)} \qquad (12\text{-}2)$$

其中根号中的内容实际上是均值的二阶矩μ_2。一般的，均值m的n阶矩表示为

$$\mu_n(z)=\sum_{i=0}^{L-1}(z_i-m)^n p(z_i) \qquad (12\text{-}3)$$

（3）平滑度：纹理亮度的相对平滑度度量，对于灰度一致的区域，平滑度R等于1；对于灰度级的值有着较大差异的区域，R等于0。

$$R=\frac{1}{(1+\sigma^2)} \qquad (12\text{-}4)$$

（4）三阶矩：直方图偏斜性的度量。对于对称的直方图，此值为0。若为正值，则直方图向右偏斜，为负值则直方图向左偏斜。

$$\mu_3=\sum_{i=0}^{L-1}(z_i-m)^3 p(z_i) \qquad (12\text{-}5)$$

（5）一致性：当区域中所有灰相等时该度量最大并由此处开始减小。

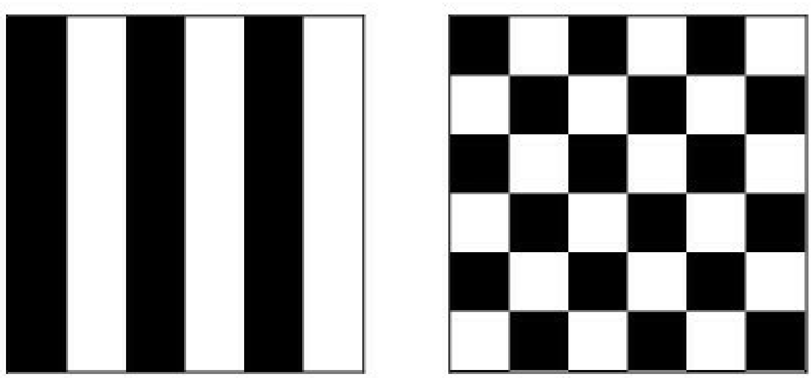
$$U=\sum_{i=0}^{L-1}p^2(z_i) \qquad (12\text{-}6)$$

（6）熵：随机性的度量。熵越大表明随机性越大，信息量也就越大；反之确定性越大，已经都确定的当然信息量就越小。这里给出熵的定量描述为

$$e=-\sum_{i=0}^{L-1}p(z_i)\log_2 p(z_i) \tag{12-7}$$

一个由均值、标准差、平滑度和熵组合而成的特征向量如v=(m,σ,R,e)。

应认识到直方图及其统计特征是一种区分能力相对较弱的特征，主要是由于直方图属于一阶统计特征，而像直方图、均值这样的一阶统计特征是无法反映纹理结构的变化的。直方图与纹理的对应关系并不是一对一的：首先，不同的纹理可能具有相同或相似的直方图，如图 12.3 所示的两种截然不同的图案就具有完全相同的直方图；其次，即便是两个不同的直方图，也可能具有相同的统计特征，如均值、标准差等。因此，依靠直方图及其统计特征来作为分类特征需要特别注意。



▲图12.3 具有相同直方图的两种图案

12.2.3 灰度共现矩阵

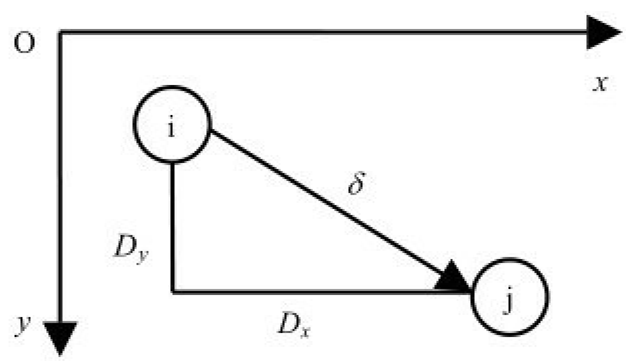
我们说灰度直方图是一种描述单个像素灰度分布的一阶统计量；而

灰度共现矩阵描述的则是具有某种空间位置关系的两个像素的联合分布，可以看成是两个像素灰度对的联合直方图，是一种二阶统计量。

1．理论基础

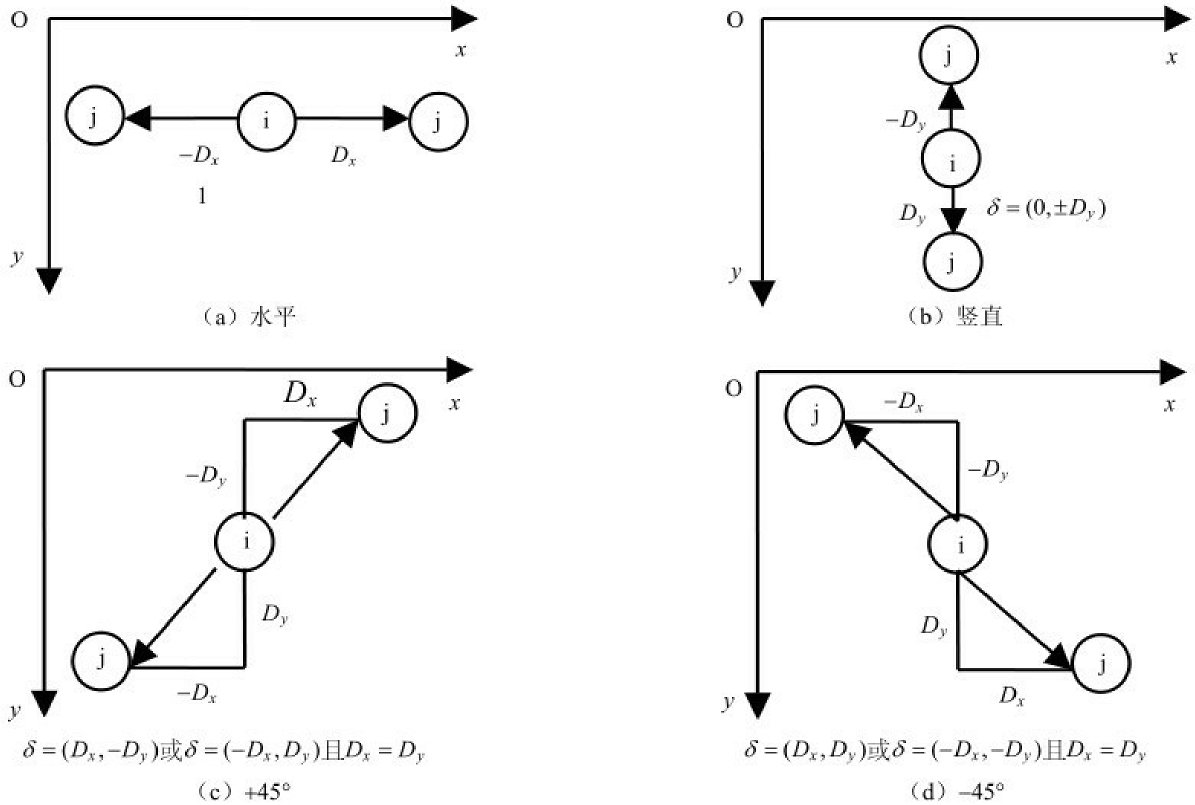
纹理是由灰度分布在空间位置上反复交替变化而形成的，因此在图像中具有某种空间位置关系的两个像素之间会存在一定的灰度关系，这种关系被称为图像灰度的空间相关特性。作为一种灰度的联合分布，灰度共现矩阵能够较好地反映这种灰度空间相关性。

我们通常用p_δ表示灰度共现矩阵，如果灰度级为L，则p_δ为一个L×L的方阵，其中的某个元素$p_\delta(i, j)$,(i, j = 0,1,2,…,L-1)被定义为具有空间位置关系δ=(D_x,D_y)，并且灰度分别为i和j的两个像素出现的次数或概率（归一化），如图12.4所示。



▲图12.4 具有空间位置关系δ且灰度分别为 i和 j的两个像素

常用的空间位置关系δ有水平、竖直和±45°，共4种，如图12.5所示。



▲图12.5 灰度共现矩阵常用的4种空间位置关系δ

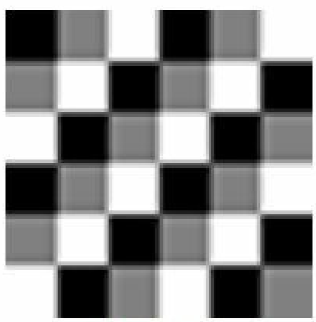
一旦空间位置关系δ确定后，就可以生成一定δ下的灰度共现矩阵p_δ，例如对于如图12.6所示的纹理，其距离为1的水平和+4°灰度共现矩阵如下所示。

$$P_\delta=\begin{bmatrix}0 & 10 & 10\\ 10 & 0 & 10\\ 10 & 10 & 0\end{bmatrix}$$，δ=(±1,0)，相应的归一化形式为

$$P_\delta=\begin{bmatrix}0 & 1/6 & 1/6\\ 1/6 & 0 & 1/6\\ 1/6 & 1/6 & 0\end{bmatrix}$$

$$P_\delta = \begin{bmatrix} 16 & 0 & 0 \\ 0 & 16 & 0 \\ 0 & 0 & 18 \end{bmatrix}$$，δ=(1,−1)orδ=(−1,1)，相应的归一化形式为

$$P_\delta = \begin{bmatrix} 8/25 & 0 & 0 \\ 0 & 8/25 & 0 \\ 0 & 0 & 9/25 \end{bmatrix}$$



（a）纹理图像的放大显示（每方格 1 像素）

0	1	2	0	1	2
1	2	0	1	2	0
2	0	1	2	0	1
0	1	2	0	1	2
1	2	0	1	2	0
2	0	1	2	0	1

（b）图（a）对应的像素灰度矩阵

▲图12.6 图像 littleTexture.bmp的纹理像素示意图

由于灰度共现矩阵p_δ总共含有L×L个元素，当灰度级L比较大时它将是一个庞大的方阵。如对于一般的256灰度图，p_δ就是一个256×256的矩阵，共2^{16} 个元素。如此庞大的矩阵将使得后续的计算量剧增。因此普通灰度图像通常要经过处理以减少灰度级数，而后再计算灰度共现矩阵。可以通过分析纹理图像的直方图，在尽量不影响纹理质量的情况下，进行适当的灰度变换来达到灰度级压缩的目的。

2．Matlab实现

计算灰度共现矩阵的Matlab实现代码如下，它被封装在金羽图书论坛（http://bbs.book95.com）的“金羽图书与答疑”板块与本书同名的主题帖子附件第12章的code文件夹的grayMat函数文件中。

```
function grayMat = grayMat( I, nLevel, x1, y1, x2, y2 )
```

```
% 计算图像I的灰度共现矩阵
% I - 输入图像
% nLevel - 要统计的灰度级数目
% x1, y1, x2, y2 - 要统计的空间位置关系
%
% grayMat - 灰度共现矩阵，大小为nLevel*nLevel
if nargin < 3
   x1 = 1;% 默认是+45°的位置关系
   y1 = -1;
   x2 = -1;
   y2 = 1;
end
% 对 I 进行灰度级划分
minVal = min(I(:));
maxVal = max(I(:));
[m n] = size(I);
if (maxVal - minVal + 1) >= nLevel
span = double(maxVal-minVal) / nLevel;
ruler = double(minVal):span:double(maxVal);
ind = find(I < ruler(2));
I2 = I; % I2为进行灰度级压缩后的图像，灰度级下标从1开始
I2(ind) = 1;
clear ind;
for iLevel = 2:nLevel-1
   for ii = 1:m
      for jj = 1:n
         if ( (I(ii, jj) >= ruler(iLevel)) && (I(ii, jj) < ruler(iLevel+1)) )
```

```
            I2(ii, jj) = iLevel;
         end
      end
   end
end
ind = find(I >= ruler(nLevel));
I2(ind) = nLevel;
clear ind;
else
   error('hello');
end
grayMat = zeros(nLevel, nLevel);
% 计算灰度共现矩阵
for ii = 1:m
   for jj = 1:n
      ii2 = ii+y1;
      jj2 = jj+x1;
      if( (ii2 >= 1) && (ii2<=m) &&(jj2>=1) &&(jj2<=n) )
         grayMat(I2(ii, jj), I2(ii2, jj2)) = grayMat(I2(ii, jj), I2(ii2, jj2)) + 1;
      end
      ii2 = ii+y2;
      jj2 = jj+x2;
      if( (ii2 >= 1) && (ii2<=m) &&(jj2>=1) &&(jj2<=n) )
         grayMat(I2(ii, jj), I2(ii2, jj2)) = grayMat(I2(ii, jj), I2(ii2, jj2)) + 1;
      end
   end
end
```

下面给出一个利用grayMat函数计算灰度共现矩阵的调用示例。

```
>> I=imread('../littleTexture.bmp');
>> gm = grayMat(I, 3) %默认计算+45°位置关系的空间共现矩阵，灰度级为3
gm =
  16    0    0
  0    16    0
  0    0    18
>> gm = grayMat(I, 3, 1, 0, -1, 0) %计算水平位置关系的空间贡献值矩阵，灰度级为3
gm =
  0    10    10
  10    0    10
  10    10    0
```

同获得直方图特征向量类似，我们可以直接将矩阵 p_δ的按行或按列存储得到的向量作为特征向量。但由于p 通常较大，更多的时候我们也是 p 的某些重要的统计特征（如二阶矩、对比度和熵等）组合成为特征向量。

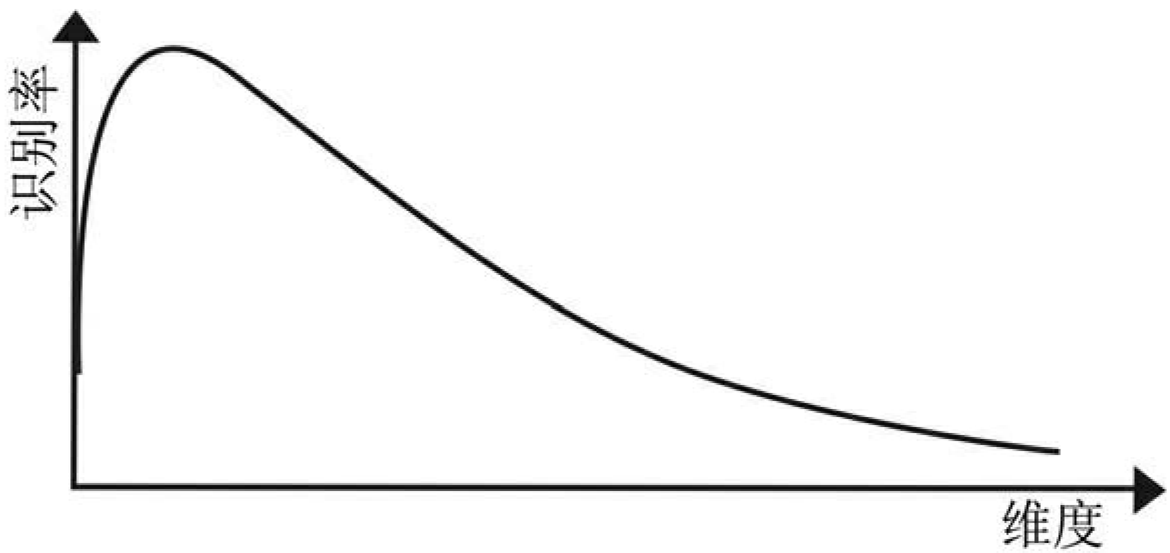
12.3 特征降维

12.3.1 维度灾难

之前已经不止一次提到了特征向量的维数过高会增加计算的复杂度，给后续的分类的问题带来负担。实际上维数过高的特征向量对于分类性能（识别率）也会造成负面的影响。直观上通常认为样本向量的维数越高，就了解了样本更多方面的属性，掌握了更多的情况，应该对提

高识别率有利。然而，事实却并非如此。

如图 12.7 所示，对于已知的样本数目，存在着一个特征数目的最大值，当实际使用的特征数目超过这个最大值时，分类器的性能不是得到改善，而是退化。这种现象正是在模式识别中被称为“维度灾难”的问题的一种表现形式。例如，我们要区分西瓜和冬瓜，表皮的纹理和长宽比例都是很好的特征，还可以再加上瓜籽的颜色以辅助判断，然而继续加入重量、体积等特征可能是无益的，甚至还会对分类造成干扰。

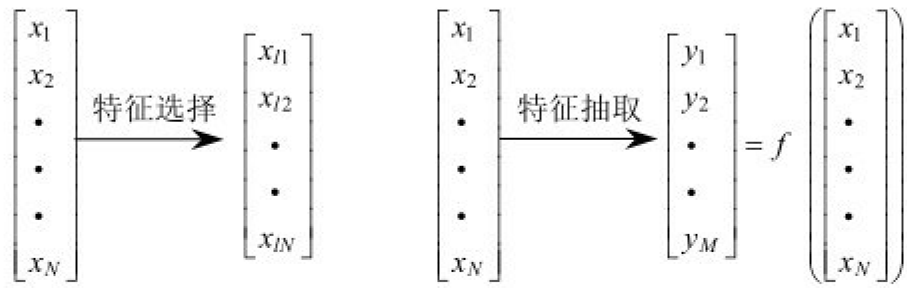


▲图12.7 识别率随特征维数的变化情况

基于以上所述的原因，降维就对我们产生了巨大的吸引力。在低维空间中计算和分类都将变得简单很多，训练（教授分类器如何区分不同类样本的过程，详见第13章）所需的样本数目也会大大降低。通过选择好的特征，摒弃坏的特征（12.3.2 特征选择），将有助于分类器性能的提升。在通过组合特征降维时，在绝大多数情况下，丢弃某些特征所损失的信息通过在低维空间中更加精确的映射（12.3.3特征抽取）可以得到补偿。

具体地说，降低维度又存在着两种方法：特征选择和特征抽取。如

图 12.8 所示，特征选择是指选择全部特征的一个子集作为特征向量；特征抽取是指通过已有特征的组合建立一个新的特征子集，12.3.2小节将要介绍的主成份分析方法（principal component analysis，PCA）就是通过原特征的线性组合建立新的特征子集的一种特征抽取方法。



▲图12.8 特征选择和特征抽取

12.3.2 特征选择简介

重新回到12.1.3小节那个鸢尾属植物的问题。对于每一个鸢尾属植物样本，总共有4个属性可以使用——花瓣长度、花瓣宽度、萼片长度和萼片宽度。我们的目的是从中选择2个属性组成特征向量用于分类这3类鸢尾属植物。

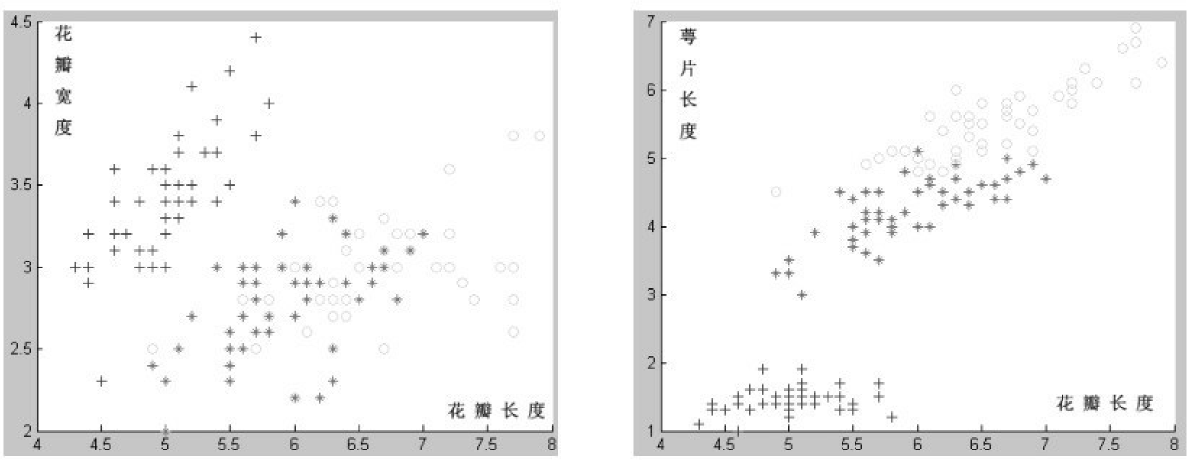
例12.2中的Matlab程序选择了不同的特征子集，并给出了在对应特征空间中样本分布的可视化表示。

[例12.2]选择不同特征对分类影响的可视化分析。

```
>> load fisheriris %载入Matlab自带的鸢尾属植物数据集
>> data = [meas(:,1), meas(:,2)]; %采用花瓣长度和花瓣宽度作为特征
>> figure
>> scatter(data(1:50, 1), data(1:50, 2), 'b+') % 第一类
hold on,scatter(data(51:100, 1), data(51:100, 2), 'r*') % 第二类
hold on,scatter(data(101:150, 1), data(101:150, 2), 'go') % 第三类
```

```
>> data = [meas(:,1), meas(:,3)]; %采用花瓣长度和萼片长度作为特征
>> figure
>> scatter(data(1:50, 1), data(1:50, 2), 'b+') % 第一类
hold on,scatter(data(51:100, 1), data(51:100, 2), 'r*') % 第二类
hold on,scatter(data(101:150, 1), data(101:150, 2), 'go') % 第三类
```

上述程序的运行结果如图12.9所示。从图12.9中可见选择不同的特征对于分类的影响有多么的严重。图12.9（a）中以‘*’和‘o’所代表的两类样本在二维空间——花瓣长度-花瓣宽度中互相交叠，很难区分；而在图12.9（b）中，选择了花瓣长度和萼片长度作为特征的情况下，相对变得容易区分了很多。由此可得出结论：对于分类3种鸢尾属植物这个模式识别问题，由花瓣长度和萼片长度组成的特征向量较由花瓣长度和花瓣宽度组成的特征向量更具区分力。



（a）采用花瓣长度和花瓣宽度作为特征　　（b）采用花瓣长度和萼片长度作为特征

▲图12.9 3类不同的鸢尾属植物的 150个样本在不同二维特征空间中的分布情况

12.3.3 主成分分析

特征抽取是指通过已有特征的组合（变换）建立一个新的特征子

集。在众多的组合方法当中，线性组合（变换）因其计算简单且便于解析分析的特点而显得颇具吸引力。下面就介绍一种通过特征的线性组合来实现降维的方法——主成分分析（Principal Conponent Analysis，PCA）。PCA的实质就是在尽可能好地代表原始数据的前提下，通过线性变换将高维空间中的样本数据投影到低维空间中。

1．理论基础

主要成份分析(PCA)是多变量分析中最老的技术之一，它来源于通信理论中的K-L变换。1901年由Pearson第一次提出主要成分分析方法，直到1963年Karhunan Loève对该问题的归纳经历了多次修改。

● 问题描述

对于d维空间中的n个样本$\vec{x}_1,\vec{x}_2,\ldots,\vec{x}_n$，考虑如何能在低维空间中最好地代表它们。

● 理论推导

（1）0维时的情况

让我们首先从“零”开始，即考虑在0维空间（一个点）中，如何以一个d维向量$\vec{x}_0$（d维空间中的一个点）来表示这n个样本，使得$\vec{x}_0$到这n个样本的距离平方和$e_0(\vec{x}_0)$最小，其中：

$$E_0(\vec{x}_0)=\sum_{i=1}^{n}\|\vec{x}_0-\vec{x}_i\|^2 \qquad (12\text{-}8)$$

以$\vec{m}$表示样本均值，即$\vec{m}=\frac{1}{n}\sum_{i=1}^{n}\vec{x}_i$，则有

$$E_0(\vec{x}_0)=\sum_{i=1}^{n}\|(\vec{x}_0-\vec{m})-(\vec{x}_i-\vec{m})\|^2$$

$$=\sum_{i=1}^{n}\|\vec{x}_0-\vec{m}\|^2-2\sum_{i=1}^{n}(\vec{x}_0-\vec{m})^t(\vec{x}_i-\vec{m})+\sum_{i=1}^{n}\|\vec{x}_i-\vec{m}\|^2$$

$$=\sum_{i=1}^{n}\|\vec{x}_0-\vec{m}\|^2-2(\vec{x}_0-\vec{m})^t\sum_{i=1}^{n}(\vec{x}_i-\vec{m})+\sum_{i=1}^{n}\|\vec{x}_i-\vec{m}\|^2$$

$$\because\sum_{i=1}^{n}(\vec{x}_i-\vec{m})=\sum_{i=1}^{n}\vec{x}_i-n\cdot\vec{m}=\sum_{i=1}^{n}\vec{x}_i-n\cdot\frac{1}{n}\sum_{i=1}^{n}\vec{x}_i=0$$

$$\therefore E_0(\vec{x}_0)=\sum_{i=1}^{n}\|\vec{x}_0-\vec{m}\|^2+\sum_{i=1}^{n}\|\vec{x}_i-\vec{m}\|^2 \quad (12\text{-}9)$$

由于第二项与$\vec{x}_0$无关，显然，$e_0(\vec{x}_0)$在$\vec{x}_0=\vec{m}$时取得最小值。这一结论表明能够在最小均方意义下最好地代表原来的n个样本的 d 维向量就是这n个样本的均值。换言之，如果只允许以d维空间中的一个点作为d维空间中原始n个样本点的代表，这个点就是这n个样本点的均值。

（2）1维时的情况

样本均值是样本数据集的零维表达（1 个点）。它非常简单，但所有样本在零维的空间中都被压缩到了同一个点，因此无法反映出样本之间的差异，也就无法进行分类。为使样本具有可分区性，进一步考虑一维（d维空间中的1条直线）的情况，通过把全部样本向通过样本均值G $\vec{m}$的1条直线作垂直投影，能够得到全部样本的一维表达。令$\vec{e}$表示这条通过均值的直线的单位方向向量，则直线方程可表示为

$$\vec{x}=\vec{m}+a\vec{e}$$

其中a为一个实数的标量，表示直线上某个点离开点m的距离。

如图12.10所示，我们以样本$\vec{x}_i$在直线上$\vec{e}$ 的垂直投影A_i作为$\vec{x}_i$的一维表达，记作$\vec{x}_i^{(1)}$ =(A_i)，上角标(1)表示是在1维空间中的表示。而

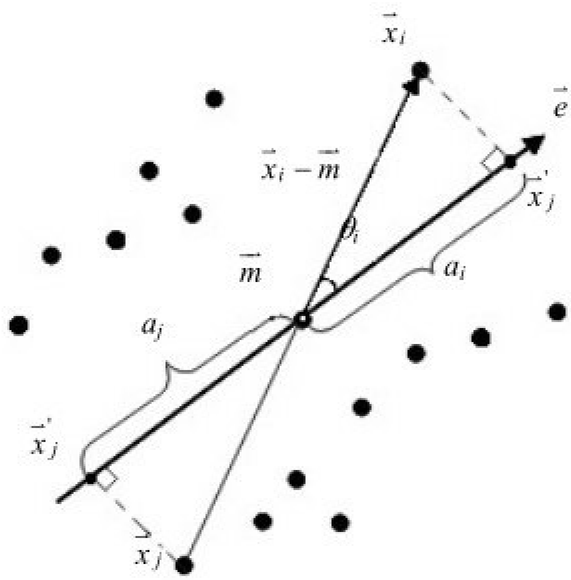
$\vec{x}_i' = \vec{m} + a_i\vec{e}$可以看作是在一维空间（直线$\vec{e}$）中 $\vec{x}_i$的近似，由垂直关系可知

$$a_i = |\vec{x}_i - \vec{m}| \cdot \cos(\theta_i) \qquad （12\text{-}10）$$

由于θ_i是向量$\vec{x}_i - \vec{m}$与向量 $\vec{e}$ 的夹角，且$|\vec{e}_i|=1$，故上式可表示为

$$a_i = |\vec{x}_i - \vec{m}| \cdot |\vec{e}| \cdot \cos(\theta_i) = \vec{e} \bullet (\vec{x}_i - \vec{m}) = (\vec{e})^{\mathrm{T}} (\vec{x}_i - \vec{m}) \qquad （12\text{-}11）$$

其中$(\vec{e})^T$表示$\vec{e}$的转置。



▲图12.10 样本的一维表达

关键的问题就是如何确定直线$\vec{e}$的最优方向以使得使得平方误差$e_1(\vec{e})$最小。下面给出推导过程。

$$E_1(\vec{e})=\sum_{i=1}^{n}\|(\vec{m}+a_i\vec{e})-\vec{x}_i\|^2=\sum_{i=1}^{n}\|a_i\vec{e}-(\vec{x}_i-\vec{m})\|^2$$

$$=\sum_{i=1}^{n}a_i^{\ 2}\|\vec{e}\|^2-2\sum_{i=1}^{n}a_i\vec{e}^T(\vec{x}_i-\vec{m})+\sum_{i=1}^{n}\|\vec{x}_i-\vec{m}\|^2$$

将式（12-11）代入，得

$$E_1(\vec{e})=\sum_{i=1}^{n}a_i^{\ 2}-2\sum_{k=i}^{n}a_i^{\ 2}+\sum_{i=1}^{n}\|\vec{x}_i-\vec{m}\|^2$$

$$=-\sum_{i=1}^{n}a_i^{\ 2}+\sum_{i=1}^{n}\|\vec{x}_i-\vec{m}\|^2$$

$$=-\sum_{i=1}^{n}(\vec{e}^T(\vec{x}_i-\vec{m}))^2+\sum_{i=1}^{n}\|\vec{x}_i-\vec{m}\|^2$$

$$=-\sum_{i=1}^{n}\vec{e}^{\mathrm{T}}(\vec{x}_i-\vec{m})(\vec{x}_i-\vec{m})^{\mathrm{T}}\vec{e}+\sum_{i=1}^{n}\|\vec{x}_i-\vec{m}\|^2$$

$$=-\vec{e}^{\mathrm{T}}(\sum_{i=1}^{n}(\vec{x}_i-\vec{m})(\vec{x}_i-\vec{m})^{\mathrm{T}})\vec{e}+\sum_{i=1}^{n}\|\vec{x}_i-\vec{m}\|^2$$

$$=-\vec{e}^{\mathrm{T}}S\vec{e}+\sum_{i=1}^{n}\|\vec{x}_i-\vec{m}\|^2 \qquad \text{（12-12）}$$

其中，d×d矩阵$S=\sum_{i=1}^{n}(\vec{x}_i-\vec{m})(\vec{x}_i-\vec{m})^T$，称为散布矩阵（scatter matrix），观察它的形式不难发现，散布矩阵S实际上是样本协方差矩阵的n−1倍。

在式（12-12）中，第二项与$\vec{e}$无关，显然，要想让$e_1(\vec{e})$最小，就要使第一个负号项$\vec{e}^{\mathrm{T}}S\vec{e}$最大。$\vec{e}^{\mathrm{T}}S\vec{e}$的最大化是一个带有约束条件的优化问题，可以采用在高等数学中的拉格朗日乘数法求解，约束条件为|$\vec{e}$

|=1。

令$y=\vec{e}^{\mathrm{T}}S\vec{e}-\lambda(\vec{e}^{\mathrm{T}}\vec{e}-1)$，其中λ为拉格朗日乘数，通过对$\vec{e}$求偏导并令偏导为0，得

$$\frac{\partial y}{\partial \vec{e}}=2S\vec{e}-2\lambda\vec{e}=0 \Rightarrow S\vec{e}=\lambda\vec{e} \qquad (12\text{-}13)$$

式（12-13）的推导中用到了矩阵运算的结论$\frac{\partial \vec{e}^{\mathrm{T}}S\vec{e}}{\partial \vec{e}}=(S+S^T)\vec{e}=0$，因散布矩阵S为对称阵，故有

$$\frac{\partial \vec{e}^{T}S\vec{e}}{\partial \vec{e}}=2S\vec{e} \qquad (12\text{-}14)$$

式（12-13）中S为一个d阶方阵，$\vec{e}$是一个d维向量，λ为一个实数。显然，这是线性代数中本征方程的典型形式，λ是本征值，而$\vec{e}$是散布矩阵S的本征向量。对式（12-13）稍加变形，两边同时左乘$\vec{e}^{T}$，得

$$\vec{e}^{T}S\vec{e}=\lambda\vec{e}^{T}\vec{e}=\lambda \qquad (12\text{-}15)$$

至此，可以很自然地得出结论：为了最大化$\vec{e}^{\mathrm{T}}S\vec{e}$，应当选取散布矩阵S的最大本征值所对应的本征向量作为投影直线$\vec{e}$的方向。也就是说，通过将全部n个样本$\vec{x}_1,\vec{x}_2,\ldots,\vec{x}_n$向以散布矩阵最大本征值对应的本征向量为方向的直线投影，可以得到最小平方误差意义下这n个样本的一维表示$A_1,A_2,\ldots,A_n$。从本质上来说，这个投影变换实际上就是基的转换。在原来的d维空间中，d个基分别是每个坐标轴方向的单位矢量$\vec{\phi}_i$ (i=1,2,…,n)，空间中的某个样本$\vec{x}_i=(\vec{x}_{i1},\vec{x}_{i2},\cdots,\vec{x}_{id})$可以由这组基表示为

$$\vec{x}_i = \sum_{k=1}^{d} x_{ik}\vec{\phi}_i \qquad (12\text{-}16)$$

投影至直线$\vec{e}$之后，在新的一维空间中（一条直线），单位矢量$\vec{e}$成为了唯一的1个基，那么在这个一维空间中的某个样本$\vec{x}_i'$同样可以由这个基向量$\vec{e}$表示为

$$\vec{x}_i' = \vec{m} + a_i\vec{e} \qquad (12\text{-}17)$$

此时$\vec{x}_i'$就是原始样本$\vec{x}_i$经过投影变换降维后的一维描述。注意到在原来的d维空间中我们是以某个样本$\vec{x}_i$的基展开式式（12-16）中基的系数来表示此样本$\vec{x}_i=(\vec{x}_{i1}, \vec{x}_{i2}, \cdots, \vec{x}_{id})$。同样在一维空间中我们以基$\vec{e}$的系数将它表示为一维向量$\vec{x}_i^{(1)}$=($A_i$)。

（3）推广至d'维

上述的结论可以立刻从一维空间（直线）的投影推广至 d'维空间（d'-d）的投影。我们将式（12-17）重写为

$$\vec{x} = \vec{m} + \sum_{k=1}^{d'} a_k\vec{e}_k \qquad (12\text{-}18)$$

新的平法误差准则函数为

$$E_{d'}(\vec{e}_1, \vec{e}_2, ..., \vec{e}_{d'}) = \sum_{i=1}^{n} \| (\vec{m} + \sum_{k=1}^{d'} a_{ik}\vec{e}_i) - \vec{x}_i \|^2$$

容易证明$e_{d'}$在向量$\vec{e}_1$,$\vec{e}_2$,…,$\vec{e}_{d'}$分别为散布矩阵S的前d'个（从大到小）本征值所对应的本征向量时取得最小值。因为散布矩阵S为实对称矩阵，因此这些本征向量都是彼此正交的。这些本征向量$\vec{e}_1$,$\vec{e}_2$,…,$\vec{e}_{d'}$就构成了在低维空间中（d'维）中的一组基向量，任何一个属于此d'维

空间的向量$\vec{x}_i'$均可由这组基表示。

$$\vec{x}_i' = \vec{m} + \sum_{k=1}^{d'} a_{ik}\vec{e}_k \qquad (12\text{-}19)$$

其中，

$$a_{ik} = \vec{e}_k \bullet (\vec{x}_i - \vec{m}) = \vec{e}_k{}^T(\vec{x}_i - \vec{m}) \qquad (12\text{-}20)$$

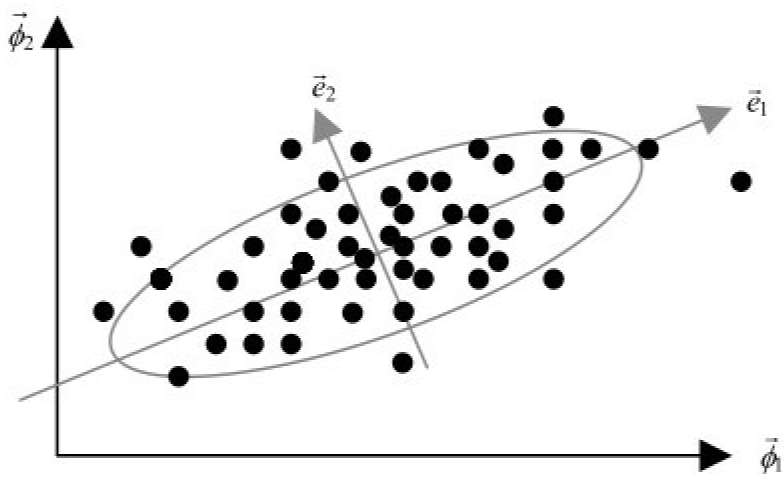
式（12-19）中对应于基$\vec{e}_1$,$\vec{e}_2$,…,$\vec{e}_{d'}$的系数A_{i1},A_{i2},…,$A_{id'}$被称作主成份(principalcomponent)， d’维向量$\vec{x}_i^{(d')}$=(A_{i1},A_{i2},…,$A_{id'}$)即为原样本$\vec{x}_i$在由基向量$\vec{e}_1$,$\vec{e}_2$,…,$\vec{e}_{d'}$所张成的d'维空间中的低维表示，而$\vec{x}_i'$实际上是对原样本$\vec{x}_i$的一种近似，且近似的程度随着d’的增大而增加，这一过程可以看作是对原样本$\vec{x}_i$的重建。在这个意义上，我们常将式（12-20）通过投影计算系数A_{ik}的过程称为分解，而将式（12-19）在变换空间中计算$\vec{x}_i'$的过程称为重构。

注意到上面这些内容和傅立叶变换中的分解（傅立叶变换）与重构（傅立叶反变换）十分相似，只是傅立叶变换中的基是sin和cos形式的基函数，而这里的基则是基矢量（向量）。两者从本质上来说都是一种基或者说是坐标系的变换，分解过程就是要将原始数据或函数转换到新的基（坐标系）下，而重建过程就是利用新的基来表示原始数据或函数。

2．几何解释

从多元统计分析的角度来看，样本$\vec{x}_1$,$\vec{x}_2$,…,$\vec{x}_n$在原d维空间中形成了一个d维椭球形状的云团，而散布矩阵的本征向量就是这个椭球的主轴，如图12.11所示。PCA实际上是寻找云团散布最大的那些主轴方向，通过向这些方向向量所张成的空间的投影达到了对特征空间降维的目的。同时，从图12.11中不难发现，PCA投影转换坐标系（从原来的

$\vec{\phi}_1$ - $\vec{\phi}_2$坐标系转换为$\vec{e}_1$-$\vec{e}_2$坐标系）的过程实际上也是去除数据线性相关性的过程，这一点读者可以通过计算PCA变换前后样本的协方差矩阵来进行验证。



▲图12.11 二维空间中的样本椭球云团及其 2个主轴

3．PCA计算

在前面略显枯燥的理论知识之后，让我们来看一个PCA计算的实例，旨在帮助读者巩固之前的PCA理论并掌握PCA计算的要点。

[例12.3]主成分计算。

计算下面两维数据集合的主要成份分量，并利用PCA方法将数据降至一维和二维。然后尝试利用1个和2个主成份实现对第一个样本的重构。

$$X = \{(1,2),(3,3),(3,5),(5,4),(5,6),(6,5),(8,7),(9,8)\}。$$

（1）计算散布矩阵S的本正向量

首先计算散布矩阵S（或样本协方差矩阵，n-1倍的系数不会影响本征向量的计算）：

样本均值：$\vec{m}$ =(5,5)

$$S=\sum_{i=1}^{8}(\vec{x}_i-\vec{m})(\vec{x}_i-\vec{m})^T=\begin{bmatrix}6.25 & 4.25\\ 4.25 & 3.5\end{bmatrix}$$

接下来，我们解式（12-13）中的本征方程。

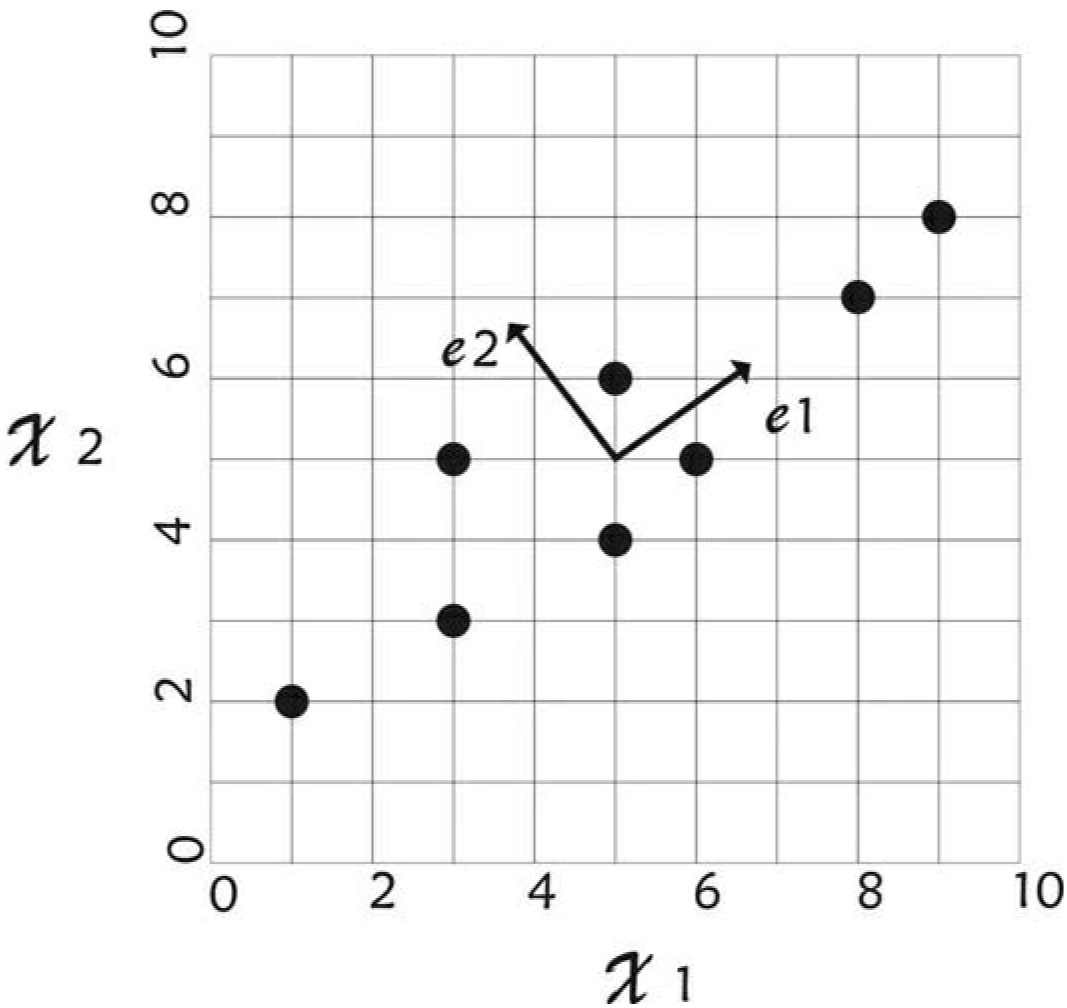
$S\vec{e}=\lambda\vec{e}\Rightarrow|S-\lambda I|\vec{e}=0$，其中I为2×2的单位矩阵

$$\Rightarrow\begin{vmatrix}6.25-\lambda & 4.25\\ 4.25 & 3.5-\lambda\end{vmatrix}=0$$

展开上式左侧的行列式，最终解得

$$\lambda_1=9.34,\lambda_2=0.41$$

再将λ_1和λ_2分别代入式（12-13），解得：$\vec{e}_1$ =(0.81,0.59)^T，$\vec{e}_2$ = (-0.59,0.81)^T。注意$\vec{e}_1$和$\vec{e}_2$是彼此正交的单位向量，如图12.12所示。



▲图12.12 样本分布及其主轴方向

（2）降至一维

通过将8个样本向其主轴$\vec{e}_1$投影，可以得到这8个样本点的一维表示。

根据公式：$a_{i1} = \vec{e}_1^{\,T}(\vec{x}_i - \vec{m})$,i=1,2,…,8，得

$$a_{11} = -5.01, a_{21} = -2.8, a_{31} = -1.62, a_{41} = -0.59, a_{51} = 0.59, a_{61} = 0.81,$$
$$a_{71} = 3.61, a_{81} = 5.01$$

从而1维表示为

$\vec{x}_1^{(1)}=-5.01, \vec{x}_2^{(1)}=-2.8, \vec{x}_3^{(1)}=-1.62, \vec{x}_4^{(1)}=-0.59, \vec{x}_5^{(1)}=0.59, \vec{x}_6^{(1)}=0.81,$
$\vec{x}_7^{(1)}=3.61, \vec{x}_8^{(1)}=5.01$

（3）降至二维

类似的，再向主轴$\vec{e}_2$投影，根据$a_{i2}=\vec{e}_2{}^T(\vec{x}_i-\vec{m})$,i=1,2,…,8可得

$a_{12}=-0.07, a_{22}=-0.44, a_{32}=1.18, a_{42}=-0.81, a_{52}=0.81, a_{62}=-0.59,$
$a_{72}=-0.15, a_{82}=0.07$

从而2维表示为

$\vec{x}_1^{(2)}=(-5.01,-0.07), \vec{x}_2^{(2)}=(-2.8,-0.44), \vec{x}_3^{(2)}=(-1.62,1.18), \vec{x}_4^{(2)}=(-0.59,-0.81)$，
$\vec{x}_5^{(2)}=(0.59,0.81), \vec{x}_6^{(2)}=(0.81,-0.59), \vec{x}_7^{(2)}=(3.61,-0.15), \vec{x}_8^{(2)}=(5.01,0.07)$。

（4）重构

如果仅利用第1个主成分分量实现对样本$\vec{x}_1$的近似（重构），则

$$\vec{x}_1{}'=\vec{m}+a_{11}\vec{e}_1=(0.9419,\ 2.0441)$$

近似程度可以用$\vec{x}_i{}'$与原样本$\vec{x}_1$的欧氏距离来衡量。

$$\text{dist}^{(1)}=\|\vec{x}_1-\vec{x}_1{}'\|=\sqrt{(1-0.9419)^2+(2-2.0441)^2}=0.0729$$

利用2个主成份分量实现对样本$\vec{x}_1$的近似（重构），得

$$\vec{x}_1{}'=\vec{m}+\sum_{k=1}^{2}a_{1k}\vec{e}_k=(0.983\ \ 2,1.987\ \ 4)$$

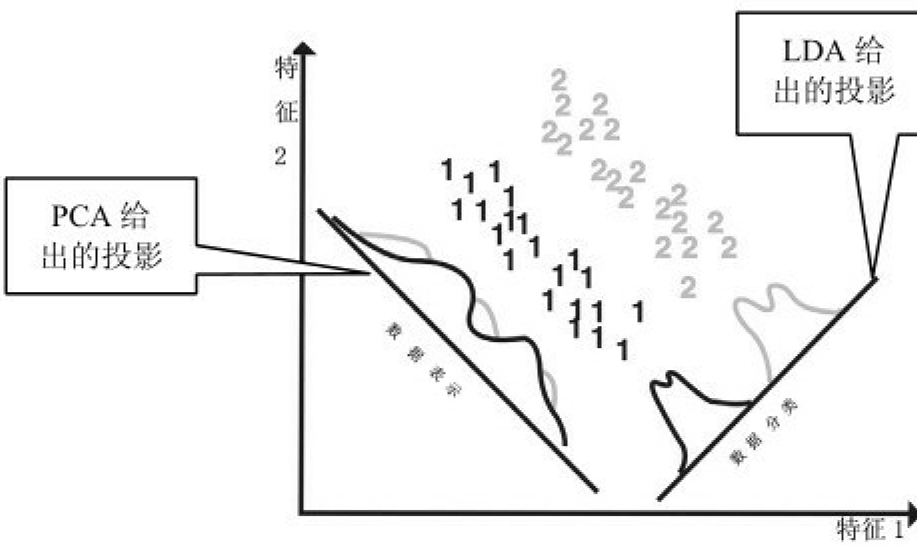
而$\text{dist}^{(2)}=\|\vec{x}_1-\vec{x}_1{}'\|=0.021$

注意到$\text{dist}^{(2)} \le \text{dist}^{(1)}$，这和之前给出的近似程度随着主成份分量数目

增大而增加的结论是一致的。

4．数据表示与数据分类

通过PCA降维后的数据并不一定是最有利于分类的，因为PCA的目的是在低维空间中尽可能好地表示原数据，确切地说是在最小均方差意义下最能代表原始数据。而这一目的有时会和数据分类的初衷相违背。图12.13说明了这种情况，PCA投影后数据样本得到了最小均方意义下的最好保留，但在降维后的一维空间中两类样本变得非常难以区分。图12.13中还给出了一种适合于分类的投影方案，对应着另一种常用的降维方法——线性判别分析（linear discriminant analysis, LDA）。PCA寻找的是能够有效表示数据的主轴方向，而LDA则是寻找用来有效分类的投影方向。



▲图12.13 分别以信号表示和分类为目的的降维

5．**PCA的Matlab**实现

函数princomp实现了对PCA的封装，其常见调用形式为

[COEFF, SCORE,latent] = princomp(X);

参数说明

● X为原始样本组成n×d的矩阵，其每一行是一个样本特征向量，每一列表示样本特征向量的一维。如对于例12.2中的问题，X就是一个8×2的样本矩阵，总共8个样本，每个样本二维。

返回值

●COEFF：主成分分量，即变换空间中的那些基向量$\vec{e}_1,\vec{e}_2,\ldots,\vec{e}_d$。也是样本协方差矩阵的本征向量。

● SCORE：主成分，X的低维表示，即X中的数据在主成分分量上的投影（可根据需要取前面几列的），也就是前文描述中的系数$A_{i1},A_{i1},\ldots,A_{id'}$。

● latent：一个包含着样本协方差矩阵本征值的向量。

作为一个调用实例，我们利用princomp函数来重新完成例12.3，相应代码如下。

```
>> X = [1,2;3,3;3,5;5,4;5,6;6,5;8,7;9,8] % 样本矩阵，每行一个样本向量
X =
  1    2
  3    3
  3    5
  5    4
  5    6
  6    5
  8    7
  9    8
>> [COEFF, SCORE, latent] = princomp(X); % 主成分分析
>> COEFF % 主成份分量（每列为一个变换空间中的基向量）
COEFF =
```

0.8086　-0.5883
0.5883　0.8086

>> SCORE % 主成份，SCORE(:, 1)为X的一维表示，SCORE为X在变换空间中的二维表示

SCORE =

-4.9995　-0.0728
-2.7939　-0.4407
-1.6173　1.1766
-0.5883　-0.8086
0.5883　0.8086
0.8086　-0.5883
3.6025　-0.1476
4.9995　0.0728

>> latent % X样本协方差矩阵的本征值

latent =

10.6764
0.4664

12.3.4 快速PCA及其实现

PCA的计算中最主要的工作量是计算样本协方差矩阵的本征值和本征向量。设样本矩阵X大小为n×d（n个d维样本特征向量），则样本散布矩阵（协方差矩阵）S将是一个d×d的方阵，故当维数d较大时计算复杂度会非常高。例如当维数 d=10 000，S是一个 10 000×10 000的矩阵，此时如果采用上面的princomp函数计算主成份，Matlab通常会出现内存耗尽的错误，即使有足够多的内存，要得到S的全部本征值可能也要花费数小时的时间。

1．理论基础

幸运的是，对于这样的问题我们并非束手无策。有一个非常好的PCA加速技巧可以用来计算矩阵S非零本征值所对应的本征向量。设$z_{n\times d}$为样本矩阵X中的每个样本减去样本均值$\vec{m}$后得到的矩阵，则散布矩阵S为$(Z^TZ)_{d\times d}$。现在考虑矩阵$R=(ZZ^T)_{n\times n}$，一般情况下由于样本数目n远远小于样本维数d，R的尺寸也远远小于散布矩阵S，然而，它与S有着相同的非零本征值。

设n维列向量$\vec{v}$是R的本征向量，则有

$$(ZZ^T)\vec{v}=\lambda\vec{v} \tag{12-21}$$

式（12-21）两边同时左乘Z^T，并应用矩阵乘法的结合律得

$$(Z^TZ)(Z^T\vec{v})=\lambda(Z^T\vec{v}) \tag{12-22}$$

式（12-22）说明$Z^T\vec{v}$为散布矩阵$S=(Z^TZ)_{d\times d}$的特征值。这说明可以计算小矩阵$R=(ZZ^T)_{n\times n}$的本征向量$\vec{v}$，而后通过左乘Z^T得到散布矩阵$S=(Z^TZ)_{d\times d}$的本征向量$Z^T\vec{v}$。

2．Matlab实现

我们编写了fastPCA函数用来对样本矩阵A进行快速主成分分析和降维（降至k维），其输出pcaA为降维后的k维样本特征向量组成的矩阵，每行一个样本，列数k为降维后的样本特征维数，相当于princomp函数中的输出SCORE，而输出V为主成分分量，即princomp函数中的输出COEFF。

fastPCA函数的实现代码如下。

```
function [pcaA V] = fastPCA( A, k )
% 快速PCA
```

```
%
% 输入：A --- 样本矩阵，每行为一个样本
%        k --- 降维至 k 维
%
% 输出：pcaA --- 降维后的 k 维样本特征向量组成的矩阵，每行一个样本，列数 k 为降维后的样本特征维数
%        V --- 主成分向量
[r c] = size(A);
% 样本均值
meanVec = mean(A);
% 计算协方差矩阵的转置 covMatT
Z = (A-repmat(meanVec, r, 1));
covMatT = Z * Z';
% 计算 covMatT 的前 k 个本征值和本征向量
[V D] = eigs(covMatT, k);
% 得到协方差矩阵 (covMatT)' 的本征向量
V = Z' * V;
% 本征向量归一化为单位本征向量
for i=1:k
    V(:,i)=V(:,i)/norm(V(:,i));
end
% 线性变换（投影）降维至 k 维
pcaA = Z * V;
% 保存变换矩阵 V 和变换原点 meanVec
save('Mat/PCA.mat', 'V', 'meanVec');
```

fastPCA的实现中调用了Matlab库函数eigs来计算矩阵$R=(ZZ^T)_{n\times n}$的前k个本征向量，即对应于最大的k个本征值的本征向量，其调用形式

为

[V, D] = eigs(R, k)

参数说明

R为要计算本征值和本征向量的矩阵。

k为要计算的本征向量数目。

返回值

- 输出矩阵$v_{n×k}$的每列对应1个本征向量，k个本征向量从左到右排列。
- 对角矩阵$D_{k×k}$对角线上的每个元素对应一个本征值。

在得到包含R的特征向量的矩阵V之后，为计算散布矩阵S的本征向量，只需计算ZTVd×k。此外，还应注意PCA中需要的是具有单位长度的本征向量，故最后要除以该向量的模，从而将正交本征向量归一化为单位正交本征向量。

这里建议读者找一个维数较高的数据集，分别利用princomp和fastPCA进行PCA计算并比较它们的效率。在下一节关于人脸特征抽取的高级应用实例中，我们将使用fastPCA对超过 10 000维的人脸样本矩阵进行主成分分析。

12.4 综合案例——基于PCA的人脸特征抽取

本节我们将应用PCA技术来抽取人脸特征。一幅人脸照片往往由比较多的像素构成，如果以每个像素作为一维特征，将得到一个维数非常高的特征向量，计算将十分困难；而且这些像素之间通常是具有相关性的。这样，利用PCA技术在降低维数的同时在一定程度上去除原始特征各维之间的相关性自然成为了一个比较理想的方案。

下面介绍这个应用案例的实现及其相关问题。本节后面将出现的Matlab实现代码都被封装在PCA_ORL工具箱中，读者可在金羽图书论

坛（http://bbs.book95.com）的“金羽图书与答疑”板块与本书同名的主题帖子附件第12章的code文件夹中找到。

12.4.1 数据集简介

本案例采用的数据集来自著名的ORL人脸库[6]。首先对该人脸库做一个简单的介绍。

● ORL数据库共有400幅人脸图像（40人，每人10幅，大小为112×92像素）。

● 这个数据库比较规范，大多数图像的光照方向和强度都差不多。

● 有少许表情、姿势、伸缩的变化，眼睛对得不是很准，尺度差异在10%左右。

● 并不是每个人都有所有的这些变化的图像，即有些人姿势变化多一点，有些人表情变化多一点，有些还戴有眼镜，但这些变化都并不大。

ORL人脸库中第1个人的一些人脸图像如图12.14所示。



▲图12.14 ORL中第 1个人的8幅人脸图片

正是基于ORL人脸库图像在光照，以及关键点如眼睛、嘴巴的位置等方面比较规范的特点，实验可以在该图片集上直接展开，而不是必须要进行归一化和校准等工作。读者可以在http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html上下载ORL人脸数据库。这里我们选用每个人的前5张图片作为实验的数据集，这样40个人共200幅样本图像。

12.4.2 生成样本矩阵

首先要做的是将这 200 幅人脸图像转换为向量形式，进而组成样本矩阵。函数 ReadFaces()用于完成这一任务。

ReadFaces()依次读入样本图像（假定40个人的样本图像位于“Data/ORL/”路径下，如第 18个人的10幅图像位于“Data/ORL/S18”中），然后将112×92的图像按列存储为一个10304维的行向量作为样本矩阵 FaceContainer 中的一个样本（一行），最后将样本矩阵保存至 Mat 目录下的FaceMat.mat文件。函数的具体实现如下：

```
function
[imgRow,imgCol,FaceContainer,faceLabel]=ReadFaces(nFacesPerPerson,
nPerson,
    bTest)
    % 读入ORL人脸库的指定数目的人脸前前五张(训练)
    %
    % 输入：nFacesPerPerson --- 每个人需要读入的样本数，默认值为 5
    %      nPerson --- 需要读入的人数，默认为全部 40 个人
```

%　　bTest --- bool型的参数。默认为0，表示读入训练样本（前5张）；如果为1，表示读入测试样本（后5张）

%

% 输出：FaceContainer --- 向量化人脸容器，nPerson * 10304 的 2 维矩阵，每行对应一个人脸向量

```
if   nargin==0 %default value
    nFacesPerPerson=5;%前5张用于训练
    nPerson=40;%要读入的人数（每人共10张，前5张用于训练）
    bTest = 0;
elseif nargin < 3
    bTest = 0;
end
img=imread('Data/ORL/S1/1.pgm');%为计算尺寸先读入一张
[imgRow,imgCol]=size(img);
FaceContainer = zeros(nFacesPerPerson*nPerson, imgRow*imgCol);
faceLabel = zeros(nFacesPerPerson*nPerson, 1);
% 读入训练数据
for i=1:nPerson
   i1=mod(i,10); % 个位
   i0=char(i/10);
   strPath='Data/ORL/S';
   if( i0～=0 )
     strPath=strcat(strPath,'0'+i0);
end
strPath=strcat(strPath,'0'+i1);
strPath=strcat(strPath,'/');
tempStrPath=strPath;
```

```
for j=1:nFacesPerPerson
  strPath=tempStrPath;
  if bTest == 0 % 读入训练数据
    strPath = strcat(strPath, '0'+j);
  else
    strPath = strcat(strPath, num2str(5+j));
  end
  strPath=strcat(strPath,'.pgm');
  img=imread(strPath);
  %把读入的图像按列存储为行向量放入向量化人脸容器faceContainer的对应行中
  FaceContainer((i-1)*nFacesPerPerson+j, :) = img(:)';
  faceLabel((i-1)*nFacesPerPerson+j) = i;
 end % j
end % i
% 保存人脸样本矩阵
save('Mat/FaceMat.mat', 'FaceContainer')
```

12.4.3 主成分分析

经过上面的处理，矩阵FaceContainter每一行就成了一个代表某个人脸样本的特征向量。通过主成分分析的方法可将这些10 304维的样本特征向量降至20维。这样数据集中每个人脸样本都可以由一个20维的特征向量来表示，以作为后续分类所采用的特征。在本书的第15章中，我们将在本节工作的基础上采用支持向量机（SVM）对这些20维的人脸样本进行分类，从而实现一个简单的人脸识别系统。

我们将对样本矩阵FaceContainer进行主成份分析的整个过程封装在

下面的main函数中，其参数 k 是主分量的数目，即降维至 k 维。该函数首先调用 ReadFaces 函数得到了人脸样本矩阵FaceContainer，而后利用 12.3.4 小节中的 fastPCA 算法计算出样本矩阵的低维表示 LowDimFaces 和主成份分量矩阵W并将LowDimFaces保存至Mat目录下的LowDimFaces.mat文件中。

```
function main(k)
% ORL 人脸数据集的主成分分析
%
% 输入：k --- 降至 k 维
% 定义图像高、宽的全局变量 imgRow 和 imgCol，它们在
ReadFaces 中被赋值
global imgRow;
global imgCol;
% 读入每个人的前5副图像
nPerson=40;
nFacesPerPerson = 5;
display('读入人脸数据...');
[imgRow,imgCol,FaceContainer,faceLabel]=ReadFaces(nFacesPerPerso
display('..............................');
nFaces=size(FaceContainer,1);%样本（人脸）数目
display('PCA降维...');
% LowDimFaces是200*20的矩阵, 每一行代表一张主成分脸(共40
人，每人5张)，每个脸20个维特征
% W是分离变换矩阵, 10304*20 的矩阵
[LowDimFaces, W] = fastPCA(FaceContainer, 20); % 主成分分析PCA
visualize_pc(W);%显示主成分脸
save('Mat/LowDimFaces.mat', 'LowDimFaces');
```

display('计算结束。');

通过下面的命令可以完成的对main函数的调用，将人脸样本向量降至20维。

%将工程所在文件夹PCA_ORL添加到系统路径列表

>> addpath(genpath('F:\doctor research\Matlab Work\ebook\PCA_ORL')) %换成您自己的PCA_ORL

绝对路径

>> main(20) %提取前20个主成分，即降至20维

上述命令运行后会在 Mat 目录下生成 LowDimFaces.mat 文件，其中的 200×20 维矩阵LowDimFaces是经过PCA降维后，原样本矩阵FaceContainer的低维表示。200个人脸样本所对应的每一个特征向量由原来的 10 304维变成了 20维，这就将后续的分类问题变为了一个在 20 维空间中的划分问题，大大得到了简化。

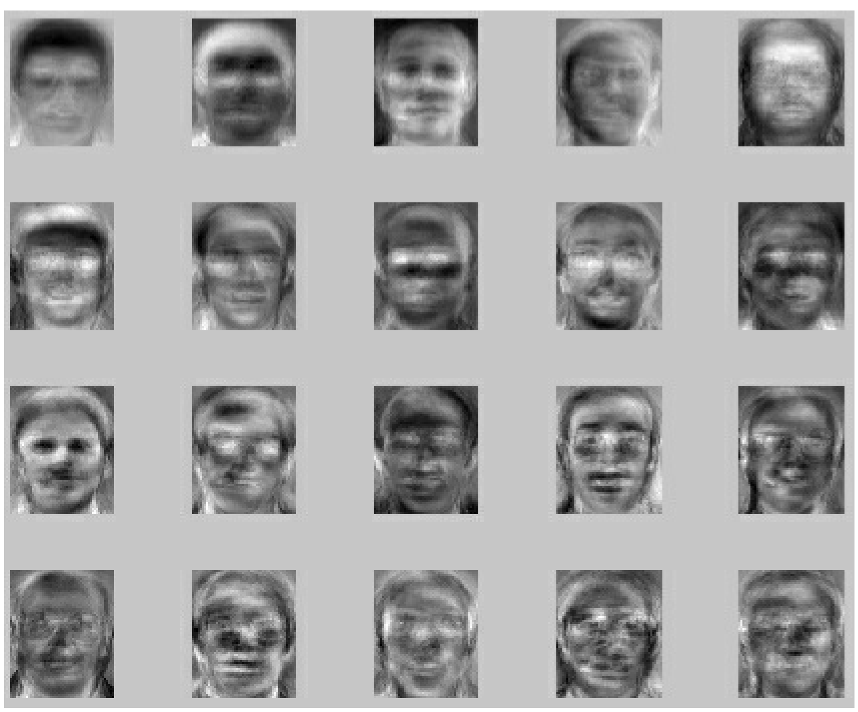
12.4.4 主成分脸可视化分析

fastPCA函数的另一个输出为主分量阵W，它是一个 10 304×20的矩阵，每列是一个 10 304维的主分量（样本协方差矩阵的本征向量），在人脸分析中，习惯称之为主成份脸。事实上我们可以将这些列向量以112×92的分辨率来显示，该工作由函数visualize_pc完成，实现如下。

```
function visualize_pc(E)
% 显示主成分分量（主成分脸，即变换空间中的基向量）
%
% 输入：E --- 矩阵，每一列是一个主成分分量
[size1 size2] = size(E);
global imgRow;
global imgCol;
```

```
row = imgRow;
col = imgCol;
if size2 ～= 20
  error('只用于显示 20 个主成份');
end;
figure
img = zeros(row, col);
for ii = 1:20
  img(:) = E(:, ii);
  subplot(4, 5, ii);
  imshow(img, []);
end
```

上述程序运行后，20个主成分脸如图12.15所示。从图12.15中不难理解为什么主分量会被称为主成份脸。



▲图12.15 20个主成分脸

1．主成分脸的定量分析

回顾式（12-20）：

$$a_{ik} = \vec{e}_k \bullet (\vec{x}_i - \vec{m}) = \vec{e}_k^{\ T} (\vec{x}_i - \vec{m})$$

我们看到，样本$\vec{x}_i$降维后的特征向量的每一维分量A_{ik}(k=1,2,…,d')都是主分量$\vec{e}_k$和个体差异脸($\vec{x}_i - \vec{m}$)的内积（（$\vec{x}_i - \vec{m}$）是某一个体人脸与平均人脸的差异）。也就是说，对于原空间中($\vec{x}_i - \vec{m}$)的每一维都与$\vec{e}_k$的对应一维相乘。因此，A_{ik}可以看成是对原d维空间中向量($\vec{x}_i - \vec{m}$)根据权向量$\vec{e}_k$的一种加权求和（线性组合）。而作为主成份脸，$\vec{e}_k$的各维分量实际上给出了在PCA降维过程中，个体差异脸($\vec{x}_i - \vec{m}$)各个维分量的重要程度。

2．结合主成分脸图像的分析

首先，我们可以看到，图12.15中的所有20个主成分脸图像的一个共同点是人脸区域之外的图像背景相对较暗，比较典型的如第1行的第3幅图像，其背景几乎为黑色，这是因为ORL数据集中的人脸图像背景较为均匀一致，在原始d维空间中样本在对应背景的这些维上差异很小，从样本分布云团上看，这些维上的云团的散布最小，因此$\vec{e}_k$对应于这些维的加权系数很小，在显现出的图像中就是灰度小，从而表现为主成份脸的暗背景。

继续按照这种思路分析，拿第1个主成分脸来说，眉毛、鼻子和上嘴唇是图像中灰度相对较高的区域，这说明实验数据集中的200个人脸之间在这些位置存在较大的差异；再比如第1行的第5个主成份脸，面部区域整体亮度较高，这可能是由数据集人脸之间的肤色差异导致的；其

他典型的如第2行的第3个以及第3行的第2个主成分脸中的眼睛，第2行第1个以及第4行第2个主成分脸的嘴，这些高亮度区域反映了实验数据集中人脸之间的五官差异。此外，我们还注意到同为五官之一的鼻子似乎并不“抢眼”，仅第3行的第1幅图像的鼻梁区域亮度相对高一些，这说明正面为主的人脸图像中鼻子之间差异不大，这正好和学术界普遍认可的鼻子在正面图像为主的人脸识别中的作用不大的结论相一致。

3．降维对分类性能的影响

人类能够识别人脸，正是由于不同的人在眼睛、嘴和眉毛等一些重要器官上的差别较大。经V=（$\vec{e}_1,\vec{e}_2,\ldots,\vec{e}_{d'}$）的线性变换后，原始d维空间中那些差别较大的维在变换至低维空间的过程中被较大的加权而保留；而那些每幅人脸图像都类似（缺乏区分力）的特征，如背景、鼻子、额头等被赋以了较低的权值，从而在d'维空间中几乎没有得到体现。这样经PCA处理后，在特征向量维数大大降低的同时，原图像中那些差异最大的特征被最大程度地保留（以一种线性组合的形式），而那些相对一致、区分力较差的特征则被丢弃。这就是为什么在很多情况下降维后，分类的识别率并不会明显下降的原因。PCA 降维丢弃某些特征所损失的信息通过在低维空间中更加精确地映射可以得到了补偿，从而可以在低维空间中得到和高维空间中相当的识别率。

4．**PCA**能够很好工作的前提

细心的读者可能会发现，在某些主成分脸图像的人脸边缘处也出现了较高的灰度，这是由数据集图像中人脸姿态和位置的差异造成的，幸好这种差异不大（10%左右）。实际上在我们的系统中，经过PCA降维后的20维样本矩阵能够很好地用于人脸识别的另一个关键点在于ORL人脸数据库中的大部分人脸在图像中占据着大致相同的区域，姿态差异度不大，并且眼睛、鼻子、发迹和嘴的位置也大体相同。否则的话，200个人脸图像之间的差异就不再是人长相本身的差异了，而是这些人的脸

部区域在图像中位置的差异、姿态的差异，以及器官位置的差异了。当然，此时PCA可以照常计算，但将降维后的样本矩阵用于人脸识别就不会取得理想的识别率了，而可能更适合于姿态分类。

12.4.5 基于主分量的人脸重建

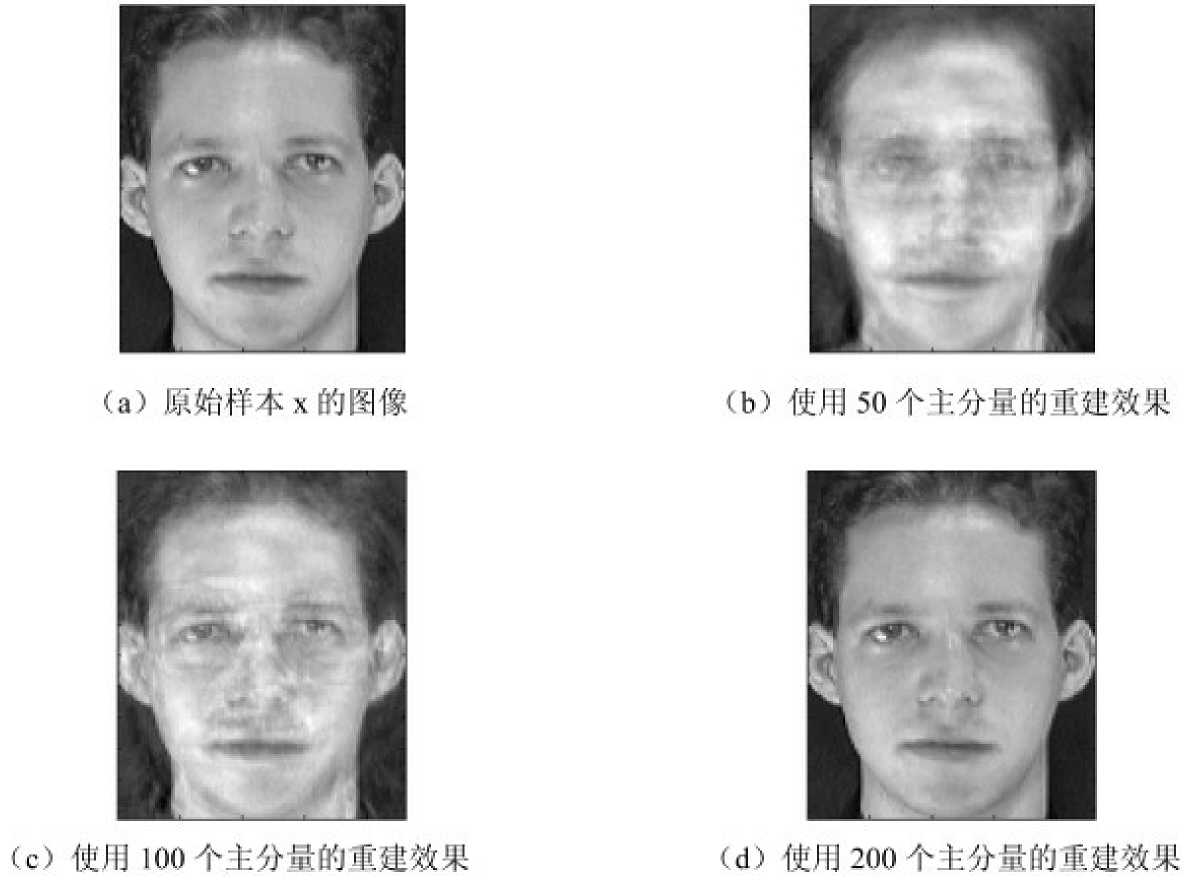
下面利用式（12-19）来实现对个体人脸图像的重建。我们提供的函数approx可以胜任这一工作。其中参数x是需要重建的个体人脸样本，k是重建使用的主分量数目，输出xApprox为对于原样本向量x的重建（近似）。具体实现如下。

```
function [ xApprox ] = approx( x, k )
% 用 k 个主成分分量来近似（重建）样本 x
%
% 输入：x --- 原特征空间中的样本，被近似的对象
%       k --- 近似（重建）使用的主分量数目
%
% 输出：xApprox --- 样本的近似（重建）
% 读入 PCA 变换矩阵 V 和 平均脸 meanVec
load Mat/PCA.mat
nLen = length(x);
xApprox = meanVec;
for ii = 1:k
  xApprox=xApprox+((x-meanVec)*V(:,ii))*V(:,ii)';
end
```

下面的程序分别采用了50、100和200个主分量来重建原始样本向量x。其中函数displayImage(x, h, w)的作用为将向量x按照h×w的分辨率进行显示。

```
>> load Mat/FaceMat.mat % 载入样本矩阵
>> x = FaceContainer(1, :); % 第一个人脸样本向量
>> displayImage(xApprox, 112 , 92); % 显示原图像
>> [pcaA V] = fastPCA( FaceContainer, 200 ); % 计算200个主分量
>> xApprox = approx(x, 50); % 使用50个主分量的近似
>> displayImage(xApprox, 112 , 92);
>> xApprox = approx(x, 100); % 使用100个主分量的近似
>> displayImage(xApprox, 112 , 92);
>> xApprox = approx(x, 200); % 使用200个主分量的近似
>> displayImage(xApprox, 112 , 92);
>> dist = norm(xApprox - x) % 计算近似的差异
dist =
   129.2606
```

上述程序最后利用norm函数计算的距离表明当使用200个主分量进行重建时，xApprox与x几乎没有差异（灰度范围为0～255的两幅112×92的图像距离仅为129）。原始图像的重建效果如图12.16所示。



▲图12.16 使用 50、100和 200个主分量对原样本的重建效果（a）和（d）实际上是同一向量在不同基下的两种不同表示

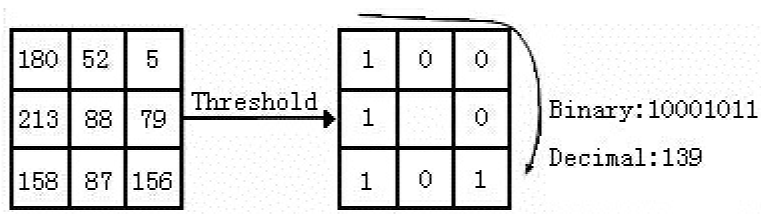
12.5 局部二进制模式

局部二进制模式（Local Binary Patterns ,LBP）最早是作为一种有效的纹理描述算子提出的，由于其对图像局部纹理特征的卓越描绘能力而获得了十分广泛的应用。LBP 特征具有很强的分类能力（Highly Discriminative）、较高的计算效率，并且对于单调的灰度变化具有不变性。

12.5.1 基本LBP

图12.17给出了一个基本的LBP算子，应用LBP算子的过程类似于滤波过程中的模板操作。逐行扫描图像，对于图像中的每一个像素点，以该点的灰度作为阈值，对其周围3×3的8邻域进行二值化，按照一定的顺序将二值化的结果组成一个8位二进制数，以此二进制数的值（0～255）作为该点的响应。

例如对于图12.17中的3×3区域的中心点，以其灰度值88作为阈值，对其8邻域进行二值化，并且从左上点开始按照顺时针方向（具体的顺序可以任意，只要统一即可）将二值化的结果组成一个二进制数10001011，即十进制的139，作为中心点的响应。在整个逐行扫描过程结束后，我们会得到一个**LBP**响应图像，这个响应图像的直方图被称为**LBP**统计直方图，或**LBP**直方图，它常常被作为我们后续识别工作的特征，因此也被称为**LBP**特征。



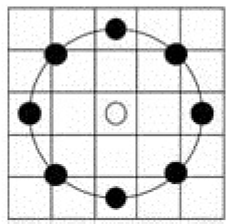
▲图12.17 基本 LBP算子

LBP 的主要思想是以某一点与其邻域像素的相对灰度作为响应，正是这种相对机制使得 LBP算子对于单调的灰度变化具有不变性。人脸图像常常会受到光照因素的影响而产生灰度变化，但在一个局部区域内，这种变化常常可以被视为是单调的，因此LBP在光照不均的人脸识别应用中也取得了很好的效果。

12.5.2 圆形邻域的LBPP,R算子

基本LBP算子可以被进一步推广为使用不同大小和形状的邻域。采

用圆形的邻域并结合双线性插值运算使我们能够获得任意半径和任意数目的邻域像素点。图12.18给出了一个半径为2的8邻域像素的圆形邻域，图中每个方格对应一个像素，对于正好处于方格中心的邻域点（左、上、右、下4个黑点），直接以该点所在方格的像素值作为它的值；对于不在像素中心位置的邻域点（斜45°方向的4个黑点），通过双线性插值确定其值。



▲图12.18 圆形（8,2）邻域的 $LBp_{8,2}$算子

这种LBP算子记作$LBp_{P,R}$，下标中P表示P邻域，R表示圆形邻域的半径。

如图12.19所示，位于图像中第i行和第j列的中心点（其灰度用 I(i, j)表示）和8个邻域点用大点标出，为计算左上角空心大黑点的值，需要利用其周围的4个像素点（4个空心小黑点）进行插值。根据4.7.2小节的双线性插值方法，首先分别计算出两个十叉点1和2的水平插值，其中点1的值根据与之处于同一行的I(i-2, j-2)以及I(i-2, j-1)的线性插值得到。

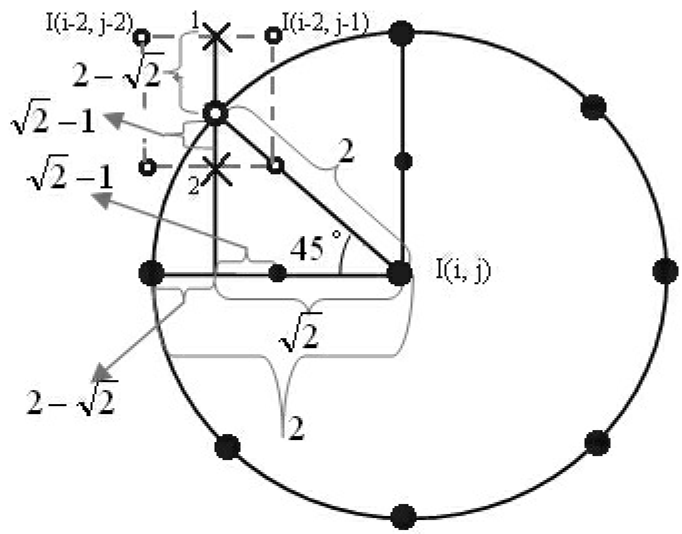
$$\text{value}(1)=I(i-2,j-2)+(2-\sqrt{2})\times(I(i-2,j-1)-I(i-2,j-2))$$

同理可计算出点2的值。

$$\text{value}(2)=I(i-1,j-2)+(2-\sqrt{2})\times(I(i-1,j-1)-I(i-1,j-2))$$

再计算出点1和点2竖直线性插值。

$$value = value(1) + (2-\sqrt{2}) \times (value(2) - value(1))$$



▲图12.19 通过双线性插值确定不在像素中心位置的邻域点（斜45°方向的四个大点）的值

12.5.3 统一化LBP算子——Uniform LBP及其Matlab实现

由于LBP直方图大多都是针对图像中的各个分区分别计算的（详见12.5.5小节），对于一个普通大小的分块区域，标准LBP算子得到的二进模式数目（LBP直方图收集箱数目）较多，而实际的位于该分块区域中的像素数目却相对较少，这将会得到一个过于稀疏的直方图，从而使直方图失去统计意义。因此应设法减少一些冗余的LBP模式，同时又保留足够的具有重要描绘能力的模式。

1．理论基础

正是基于以上考虑，研究者提出了统一化模式（uniform patterns）的概念，这是对LBP算子的又一重大改进。对于一个局部二进制模式，在将其二进制位串视为循环的情况下，如果其中包含的从0到1或者从1到0转变不多于2个，则我们称这个局部二进制模式为统一化模式（uniform patterns）。例如，模式00000000（0个转变）、01110000（2

个转变）和11001111（2个转变）都是统一化模式，而模式11001001（4个转变）和01010011（6个转变）则不是。

统一化模式的意义在于：在随后的 LBP 直方图的计算过程中，只为统一化模式分配单独的直方图收集箱（bin），而所有的非统一化模式都被放入一个公用收集箱，这就使 LBP 特征的数目大大减少。一般来说，保留的统一化的模式往往是反映重要信息的那些模式，而那些非统一化模式中过多的转变往往由随即噪声引起，不具有良好的统计意义。

假设图像分块区域大小为18×20，像素总数为360。如果采用8邻域像素的标准LBP算子，收集箱（特征）数目为256个，平均到每个收集箱的像素数目还不到2个（360/256），没有统计意义；而统一化LBP算子的收集箱数目为59（58个统一化模式收集箱加上1个非统一化模式收集箱），平均每个收集箱中将含有6个左右像素（360/59），更具统计意义。对16邻域像素而言，标准LBP算子和统一化LBP算子的收集箱数目分别为 65 536和243。

统一化LBP算子通常记作 $LBP_{P,R}^{u2}$ 。

2．Matlab实现

本节所提供的全部源代码可以在金羽图书论坛（http://bbs.book95.com）的“金羽图书与答疑”板块与本书同名的主题帖子附件的“chapter12/code/”目录下的LBP工具箱中找到，使用时可通过以下命令将工具箱添加到工作路径。

addpath(genpath('…改为您存放LBP工具箱的相应目录…\LBP '))

LBP算子具有一定的半径，类似于模板操作，这里同样要注意LBP算子应用过程中的边界问题。由于一般我们关心的是 LBP 统计直方图，而不是响应图像本身，因此实现中一般不需向外填充边界，而是直接在计算中不包括图像的边界部分。

应用 $LBP_{8,2}^{u2}$ 算子到某个分块图像并获得直方图的实现程序如下。

```
function [histLBP, MatLBP] = getLBPFea(I)
% 计算分区图像 I 的LBP特征,(8,2),uniform
%
% 输入：I --- 分区图像
%
% 返回值: MatLBP --- LBP响应矩阵
%         histLBP --- 行向量，LBP直方图
% 获得分块图像I的大小
[m n] = size(I);
rad = 2;
if (m <= 2*rad) || (n <= 2*rad)
   error('I is too small to compute LBP feature!');
end
MatLBP = zeros(m-2*rad, n-2*rad);
% 读入 LBP 映射(像素灰度与直方图收集箱索引的映射)
load Mat/LBPMap.mat;
for ii = 1+rad : m-rad
   for jj = 1+rad : n-rad
      nCnt = 1;
      % 计算(8,2)邻域的像素值，不在像素中心的点通过双线性插值获得其值
      nbPT(nCnt) = I(ii, jj-rad);
      nCnt = nCnt + 1;
      horInterp1 = I(ii-2, jj-2) + 0.5858*( I(ii-2, jj-1) - I(ii-2, jj-2) ); % 水平方向
```

插值

horInterp2 = I(ii-1, jj-2) + 0.5858*(I(ii-1, jj-1) - I(ii-1, jj-2)); % 水平方向

插值

verInterp = horInterp1 + 0.5858*(horInterp2 - horInterp1); % 竖直方向插值

nbPT(nCnt) = verInterp;

nCnt = nCnt + 1;

nbPT(nCnt) = I(ii-2, jj);

nCnt = nCnt + 1;

horInterp1 = I(ii-2, jj+1) + 0.4142*(I(ii-2, jj+2) - I(ii-2, jj+1));

horInterp2 = I(ii-1, jj+1) + 0.4142*(I(ii-1, jj+2) - I(ii-1, jj+1));

verInterp = horInterp1 + 0.5858*(horInterp2 - horInterp1);

nbPT(nCnt) = verInterp;

nCnt = nCnt + 1;

nbPT(nCnt) = I(ii, jj+2);

nCnt = nCnt + 1;

horInterp1 = I(ii+1, jj+1) + 0.4142*(I(ii+1, jj+2) - I(ii+1, jj+1));

horInterp2 = I(ii+2, jj+1) + 0.4142*(I(ii+2, jj+2) - I(ii+2, jj+1));

verInterp = horInterp1 + 0.4142*(horInterp2 - horInterp1);

nbPT(nCnt) = verInterp;

nCnt = nCnt + 1;

nbPT(nCnt) = I(ii+2, jj);

nCnt = nCnt + 1;

horInterp1 = I(ii+1, jj-2) + 0.5858*(I(ii+1, jj-1) - I(ii+1, jj-2));

horInterp2 = I(ii+2, jj-2) + 0.5858*(I(ii+2, jj-1) - I(ii+2, jj-1));

verInterp = horInterp1 + 0.4142*(horInterp2 - horInterp1);

```
        nbPT(nCnt) = verInterp;
        for iCnt = 1:nCnt
          if( nbPT(iCnt) >= I(ii, jj) )
            MatLBP(ii-rad, jj-rad) = MatLBP(ii-rad, jj-rad) + 2^(nCnt-iCnt);
          end
        end
      end
    end
    % 计算LBP直方图
    histLBP = zeros(1, 59); % 对于(8,2)的uniform直方图共有59个收集箱
    for ii = 1:m-2*rad
      for jj = 1:n-2*rad
        histLBP( vecLBPMap( MatLBP(ii, jj)+1 ) ) = histLBP(
vecLBPMap( MatLBP(ii, jj)+1 ) )
    + 1;
      end
    end
```

上述算法中围绕每一个中心点，从左侧开始，按照顺时针的顺序访问8个邻域，形成二进制模式位串。在计算直方图时借助vecLBPMap映射表将响应图像MatLBP中的像素灰度映射到其对应的收集箱编号。如灰度为gray(0≤gray≤255)的像素应落入第vecLBPMap(gray+1)号收集箱中。通过下面的函数makeLBPMap来获得映射表vecLBPMap。

```
    function vecLBPMap = makeLBPMap
    % 生成(8,2)临域uniform LBP直方图的映射关系，即将256个灰度值
映射到59个收集箱中
    % 所有的非 uniform 放入一个收集箱中
    vecLBPMap = zeros(1, 256); %初始化映射表
```

```
bits = zeros(1, 8); %8位二进制模式串
nCurBin = 1;
for ii = 0:255
  num = ii;
  nCnt = 0;
  % 获得灰度num的二进制表示bits
  while (num)
    bits(8-nCnt) = mod(num, 2);
    num = floor( num / 2 );
    nCnt = nCnt + 1;
  end
  if IsUniform(bits) % 判断bits是不是uniform模式
    vecLBPMap(ii+1) = nCurBin; % 每个uniform模式分配一个收集箱
    nCurBin = nCurBin + 1;
  else
    vecLBPMap(ii+1) = 59; %所有非uniform模式都放入第59号收集箱
  end
end
% 保存映射表
save('Mat/LBPMap.mat', 'vecLBPMap');
```

函数 makeLBPMap 中调用了 IsUniform(bits)方法来检查二进模式串 bits 是否是统一化模式（uniform pattern），IsUniform方法的实现如下。

```
function bUni = IsUniform(bits)
% 判断某一个位串模式 bits 是否是 uniform 模式
%
% 输入：bits --- 二进制LBP模式串
```

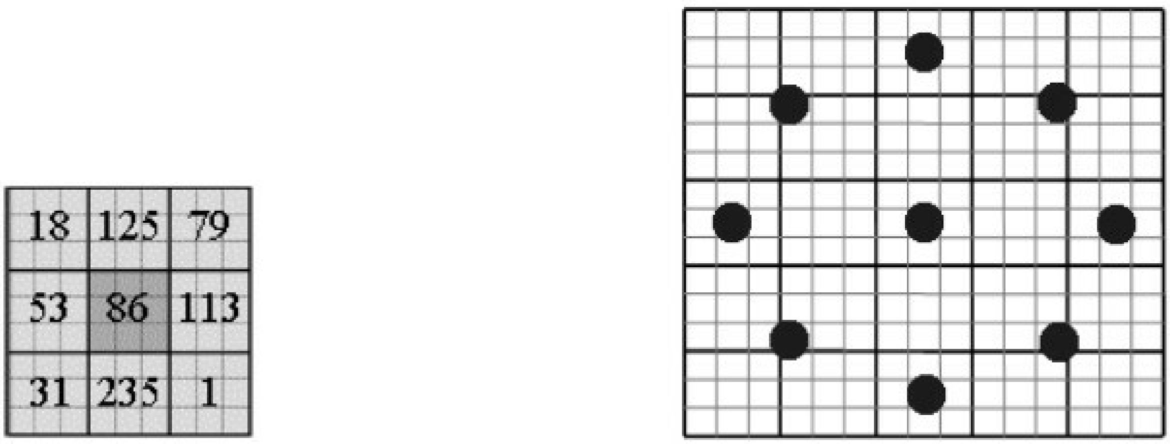
```
%
% 返回值：bUni --- =1，if bits 是uniform模式串；=2，if bits 不是uniform模式串
n = length(bits);
nJmp = 0; % 位跳变数（0->1 or 1->0）
for ii = 1 : (n-1)
  if( bits(ii) ～= bits(ii+1) )
    nJmp = nJmp+1;
  end
end
if bits(n) ～= bits(1)
  nJmp = nJmp+1;
end
if nJmp > 2
  bUni = false;
else
  bUni = true;
end
```

12.5.4 MB-LBP及其Matlab实现

1．理论基础

前述的基于像素相对灰度比较的$LBP_{P,R}^{u2}$算子可以很精细地描述图像局部的纹理信息。然而，也正是由于这种特征的局部化特点，使它易受噪声的影响而不够健壮（robust），缺乏对图像整体信息的粗粒度把握。因此MB-LBP（Multi-block Local Binary Patterns）被提出以弥补传统LBP的这一不足。起初MB-LBP被作为标准3×3 LBP的扩展而引入，随

后也被用于与$LBP_{P,R}^{u2}$算子结合使用。在MB-LBP的计算中，传统LBP算子像素值之间的比较被像素块（sub-block）之间的平均灰度的比较所代替，如图12.20所示。不同的像素块大小代表着不同的观察和分析粒度。我们通常以符号$MB_S - LBP_{8,2}^{u2}$表示像素块大小为S×S的$LBP_{8,2}^{u2}$算子。



（a） $MB_3 - LBP_{3\times3}^{u2}$　　（b） $MB_3 - LBP_{8,2}^{u2}$

▲图12.20 MB-LGBP算子

图12.20中每个灰色细线的小方格代表一个像素，每个黑色粗线围成的大方格代表一个像素块，其值是由其中3×3共9个像素的灰度平均而得的。

2．Matlab实现

下面分别给出应用$MB_S - LBP_{8,2}^{u2}$算子和$MB_3 - LBP_{3\times3}^{u2}$算子到某一图像分区I的Matlab算法实现。

（1）提取$MB_S - LBP_{8,2}^{u2}$特征

算法getMBLBPFea的输入blockSize为块的大小，即$MB_S - LBP_{8,2}^{u2}$中的S，其默认值为1，即1个块仅为1个像素，对应传统的$LBP_{8,2}^{u2}$算子。因此，在12.4.3小节的getLBPFea算法可以看作是本算法的特例。为了求得I中各个像素块的值，首先计算像素块中像素的平均灰度，而后以此平

均灰度作为灰度值，求得了I的低分辨率表示I_MB，此后的阈值化操作只需对I_MB进行，阈值化过程和getLBPFea中类似。

```
function [histLBP, MatLBP, MatLBP_MB] = getMBLBPFea(I, blockSize)
% 计算分块区域 I 的LBP特征,(8,2),uniform
%
% 输入：I --- 分区图像
%       blockSize --- MBLBP 中的分块大小，默认值为1
%
% 返回值: MatLBP --- LBP 响应矩阵
%         histLBP --- 行向量，LBP 直方图
%         MatLBP_MB --- MBLBP 的像素块低分辨率表示
if nargin < 2
  blockSize = 1;
end
% 获得分块图像I的大小
[m n] = size(I);
% 将原始图像依据 blockSize 分块，计算每块的平均灰度值，对应保存在映射矩阵 I_MB 中
mSub = floor(m / blockSize);
nSub = floor(n / blockSize);
mRem = mod(m, blockSize);
nRem = mod(n, blockSize);
mRem = round(mRem / 2);
nRem = round(nRem / 2);
I_MB = zeros(mSub, nSub);
for ii = 1:mSub
```

```
        for jj = 1:nSub
            I_center = I( 1+mRem:mRem+mSub*blockSize,
1+nRem:nRem+nSub*blockSize ); % 取中心
    区域，不够分出整块的留在两个边
            SubRgn = I_center( (ii-1)*blockSize+1 : ii*blockSize, (jj-
1)*blockSize+1 :
    jj*blockSize );
            I_MB(ii, jj) = mean( SubRgn(:) );
        end
    end
    % 剩下的任务就是对分块矩阵的映射 I_MB 计算 blockSize = 1 的
uniform (8, 2) LBP特征了
    rad = 2;
    if (mSub <= 2*rad) || (nSub <= 2*rad)
        error('I is too small to compute LBP feature!');
    end
    MatLBP_MB = zeros(mSub-2*rad, nSub-2*rad);
    % 读入 LBP 映射(像素灰度与直方图收集箱索引的映射)
    load Mat/LBPMap.mat;
    for ii = 1+rad : mSub-rad
        for jj = 1+rad : nSub-rad
            nCnt = 1;
            % 计算(8,2)邻域的像素值，不在像素中心的点通过双线性插值
获得其值
            nbPT(nCnt) = I_MB(ii, jj-rad);
            nCnt = nCnt + 1;
            horInterp1 = I_MB(ii-2, jj-2) + 0.5858*( I_MB(ii-2, jj-1) - I_MB(ii-
```

2, jj-2));

% 水平方向插值

horInterp2 = I_MB(ii-1, jj-2) + 0.5858*(I_MB(ii-1, jj-1) - I_MB(ii-1, jj-2));

% 水平方向插值

verInterp = horInterp1 + 0.5858*(horInterp2 - horInterp1); % 竖直方向插值

nbPT(nCnt) = verInterp;

nCnt = nCnt + 1;

nbPT(nCnt) = I_MB(ii-2, jj);

nCnt = nCnt + 1;

horInterp1 = I_MB(ii-2, jj+1) + 0.4142*(I_MB(ii-2, jj+2) - I_MB(ii-2, jj+1));

horInterp2 = I_MB(ii-1, jj+1) + 0.4142*(I_MB(ii-1, jj+2) - I_MB(ii-1, jj+1));

verInterp = horInterp1 + 0.5858*(horInterp2 - horInterp1);

nbPT(nCnt) = verInterp;

nCnt = nCnt + 1;

nbPT(nCnt) = I_MB(ii, jj+2);

nCnt = nCnt + 1;

horInterp1 = I_MB(ii+1, jj+1) + 0.4142*(I_MB(ii+1, jj+2) - I_MB(ii+1, jj+1));

horInterp2 = I_MB(ii+2, jj+1) + 0.4142*(I_MB(ii+2, jj+2) - I_MB(ii+2, jj+1));

verInterp = horInterp1 + 0.4142*(horInterp2 - horInterp1);

nbPT(nCnt) = verInterp;

nCnt = nCnt + 1;

```
            nbPT(nCnt) = I_MB(ii+2, jj);
            nCnt = nCnt + 1;
            horInterp1 = I_MB(ii+1, jj-2) + 0.5858*( I_MB(ii+1, jj-1) -
I_MB(ii+1, jj-2) );
            horInterp2 = I_MB(ii+2, jj-2) + 0.5858*( I_MB(ii+2, jj-1) -
I_MB(ii+2, jj-1) );
            verInterp = horInterp1 + 0.4142*( horInterp2 - horInterp1 );
            nbPT(nCnt) = verInterp;
            for iCnt = 1:nCnt
              if( nbPT(iCnt) >= I_MB(ii, jj) )
                MatLBP_MB(ii-rad, jj-rad) = MatLBP_MB(ii-rad, jj-rad) +
2^(nCnt-iCnt);
              end
            end
          end
        end
        % 还原 MatLBP_MB
        MatLBP = zeros(m-2*rad*blockSize, n-2*rad*blockSize);
        for ii = 1:mSub-2*rad
          for jj = 1:nSub-2*rad
            MatLBP( mRem+(ii-1)*blockSize+1 : mRem+ii*blockSize, nRem+
(jj-1)*blockSize+1 :
        nRem+jj*blockSize ) = MatLBP_MB(ii, jj);
          end
        end
        % 计算LBP直方图
        histLBP = zeros(1, 59); % 对于(8,2)的uniform直方图共有59个收集箱
```

```
for ii = 1:mSub-2*rad
   for jj = 1:nSub-2*rad
      histLBP( vecLBPMap( MatLBP_MB(ii, jj)+1 ) ) = histLBP(
vecLBPMap( MatLBP_MB(ii,
jj)+1 ) ) + 1;
   end
end
```

（2）提取$MB_3-LBP_{3\times3}^{u2}$特征

```
function [histLBP, MatLBP, MatLBP_MB] = getMBLBPFea_33(I,
blockSize)
% 计算分区图像 I 的LBP特征,3*3,uniform
% return value: MatLBP --- LBP 响应矩阵
%               histLBP --- 行向量，LBP 直方图
%              blockSize --- MBLBP 中的分块大小，默认值为1
%
% 输入：I --- 分区图像
%       blockSize --- 块的大小
%
% 返回值: MatLBP --- LBP 响应矩阵
%         histLBP --- 行向量，LBP 直方图
%         MatLBP_MB --- MBLBP 的像素块低分辨率表示
if nargin < 2
   blockSize = 1;
end
% 获得分块图像I的大小
[m n] = size(I);
```

% 将原始图像依据 blockSize 分块，计算每块的平均灰度值，对应保存在映射矩阵 I_MB 中

```
mSub = floor(m / blockSize);
nSub = floor(n / blockSize);
mRem = mod(m, blockSize);
nRem = mod(n, blockSize);
mRem = round(mRem / 2);
nRem = round(nRem / 2);
I_MB = zeros(mSub, nSub);
for ii = 1:mSub
  for jj = 1:nSub
    I_center = I( 1+mRem:mRem+mSub*blockSize,
1+nRem:nRem+nSub*blockSize ); % 取中心
区域，不够分出整块的留在两个边
    SubRgn = I_center( (ii-1)*blockSize+1 : ii*blockSize, (jj-
1)*blockSize+1 :
jj*blockSize );
    I_MB(ii, jj) = mean( SubRgn(:) );
  end
end
```

% 剩下的任务就是对分块矩阵的映射 I_MB 计算 blockSize = 1 的 uniform 3*3 LBP特征了

```
rad = 1;
if (mSub <= 2*rad) || (nSub <= 2*rad)
  error('I is too small to compute LBP feature!');
end
MatLBP_MB = zeros(mSub-2*rad, nSub-2*rad);
```

```
% 读入 LBP 映射(像素灰度与直方图收集箱索引的映射)
load Mat/LBPMap.mat;
for ii = 1+rad : mSub-rad
   for jj = 1+rad : nSub-rad
      nCnt = 1;
      % 计算 3*3 邻域的像素值
      nbPT(nCnt) = I_MB(ii-rad, jj-rad);
      nCnt = nCnt + 1;
      nbPT(nCnt) = I_MB(ii-rad, jj);
      nCnt = nCnt + 1;
      nbPT(nCnt) = I_MB(ii-rad, jj+rad);
      nCnt = nCnt + 1;
      nbPT(nCnt) = I_MB(ii, jj+rad);
      nCnt = nCnt + 1;
      nbPT(nCnt) = I_MB(ii+rad, jj+rad);
      nCnt = nCnt + 1;
      nbPT(nCnt) = I_MB(ii+rad, jj);
      nCnt = nCnt + 1;
      nbPT(nCnt) = I_MB(ii+rad, jj-rad);
      nCnt = nCnt + 1;
      nbPT(nCnt) = I_MB(ii, jj-rad);
      for iCnt = 1:nCnt
         if( nbPT(iCnt) >= I_MB(ii, jj) )
            MatLBP_MB(ii-rad, jj-rad) = MatLBP_MB(ii-rad, jj-rad) +
2^(nCnt-iCnt);
         end
      end
```

```
    end
end
% 还原 MatLBP_MB
MatLBP = zeros(m-2*rad*blockSize, n-2*rad*blockSize);
for ii = 1:mSub-2*rad
    for jj = 1:nSub-2*rad
        MatLBP( mRem+(ii-1)*blockSize+1 : mRem+ii*blockSize, nRem+
(jj-1)*blockSize+1 :
nRem+jj*blockSize ) = MatLBP_MB(ii, jj);
    end
end
% 计算LBP直方图
histLBP = zeros(1, 59); % 对于(8,2)的uniform直方图共有59个收集箱
for ii = 1:mSub-2*rad
    for jj = 1:nSub-2*rad
        histLBP( vecLBPMap( MatLBP_MB(ii, jj)+1 ) ) = histLBP(
vecLBPMap( MatLBP_MB(ii,
jj)+1 ) ) + 1;
    end
end
```

[例12.4]经过$MB_S-LBP_{8,2}^{u2}$滤波的人脸图像。

分别采用$MB_1-LBP_{8,2}^{u2}$、$MB_2-LBP_{8,2}^{u2}$和$MB_3-LBP_{8,2}^{u2}$算子对图12.21（a）中图像进行特征提取的Matlab程序如下所示。

```
>> I = imread('../mh_gray.bmp'); %读入图像
>> [hist1, I_LBP1] = getMBLBPFea(I, 1);
>> [hist2, I_LBP2] = getMBLBPFea(I, 2);
```

```
>> [hist3, I_LBP3] = getMBLBPFea(I, 3);
>> figure, imshow(I_LBP1, []) %得到图12.21（b）
>> figure, imshow(I_LBP2, []) %得到图12.21（c）
>> figure, imshow(I_LBP3, []) %得到图12.21（d）
```

上述程序运行后效果如图12.21（b）、（c）、（d）所示。



（a）原图像　（b）经 $MB_1-LBP_{8,2}^{u2}$ 滤波后　（c）经 $MB_2-LBP_{8,2}^{u2}$ 滤波后　（d）经 $MB_3-LBP_{8,2}^{u2}$ 滤波后

▲图12.21 经过$MB_S-LBP_{8,2}^{u2}$ 滤波的人脸图像

图12.21（b）、（c）、（d）分别为图12.21（a）经过$MB_1-LBP_{8,2}^{u2}$，$MB_2-LBP_{8,2}^{u2}$和$MB_3-LBP_{8,2}^{u2}$算子滤波之后的响应图像。注意3幅响应图像中LBP算子的强大纹理描绘能力，同时应看到随着像素块大小S的增加，响应图像中纹理增粗并且趋于稳定，说明对相对较大S的LBP直方图的分析将有助于我们把握图像中的粗粒度信息。

12.5.5 图像分区及其Matlab实现

在12.2.2小节曾提到，作为图像的一阶统计特征，直方图无法描述图像的结构信息。而图像各个区域的局部特征往往差异较大，如果仅对整个图像的生成一个 LBP 直方图，这些局部的差异信息就会丢失。分区LBP特征可有效解决这一问题。

具体的方法是将一幅图像适当地划分为P×Q个分区（partition），然后分别计算每个图像分区的直方图特征，最后再将所有块的直方图特

征连接成一个复合的特征向量（composite feature）作为代表整个图像的LBP直方图特征。

1．分区大小的选择

理论上，越小越精细的分区意味着更好的局部描述能力，但同时会产生更高维数的复合特征。然而过小的分区会造成直方图过于稀疏从而失去统计意义。在文献[7]人脸识别的应用中选择了18×21的分区大小，这可以作为对于一般问题的指导性的标准，因为它是一个精确描述能力与特征复杂度的良好折中。在表情识别中更小一些（如10×15）的分区被我们证明[8，9]能够获得更好的分类能力。这里分区大小的单位是MB-LBP的像素块（block）。如对于传统LBP，每个分区大小取18像素×21像素，则对于 $MB_3-LBP_{8,2}^{u2}$ ，分区大小应取18像素块×21像素块=54像素×63像素。

2．分区**LBP**的**Matlab**实现

我们编写了getLBPHist(I, r, c, nMB)函数用来提取图像 I的分区LBP特征。其输入 r和 c分别代表分区的行数和列数，nMB给出了MB-LBP像素块的大小。函数返回一个向量，它是图像I的复合LBP特征。

```
function histLBP = getLBPHist(I, r, c, nMB)
% 取得I的分区 LBP 直方图
%
% 输入：r,c --- 了分区的数目，r*c个分区
%       nMB --- MB-LBP 中块的大小
%
% 返回值：histLBP --- 连接 I 的各个分块的LBP直方图而形成的代表 I 的LBP复合特征向量
[m n] = size(I);
% 计算分区的大小
```

```
        mPartitionSize = floor(m / r);
        nPartitionSize = floor(n / c);
        for ii = 1:r-1
          for jj = 1:c-1
            Sub = I( (ii-1)*mPartitionSize+1:ii*mPartitionSize, (jj-
1)*nPartitionSize+1:
        jj*nPartitionSize );
            hist{ii}{jj} = getMBLBPFea( Sub, nMB ); %如需提取3*3LBP，请
注释此行
        %       hist{ii}{jj} = getMBLBPFea_33( Sub, nMB ); %如需提取
3*3LBP，请打开此注释
          end
        end
        % 处理最后一行和最后一列
        clear Sub
        for ii = 1:r-1
          Sub = I( (ii-1)*mPartitionSize+1:ii*mPartitionSize, (c-
1)*nPartitionSize+1:n );
          hist{ii}{c} = getMBLBPFea(Sub, nMB);
        %   hist{ii}{c} = getMBLBPFea_33( Sub, nMB );
        end
        clear Sub
        for jj = 1:c-1
          Sub = I( (r-1)*mPartitionSize+1:m, (jj-
1)*nPartitionSize+1:jj*nPartitionSize );
          hist{r}{jj} = getMBLBPFea(Sub, nMB);
        %   hist{r}{jj} = getMBLBPFea_33( Sub, nMB );
```

```
    end
    clear Sub
    Sub = I((r-1)*mPartitionSize+1:m, (c-1)*nPartitionSize+1:n);
    hist{r}{c} = getMBLBPFea(Sub, nMB);
    %hist{r}{c} = getMBLBPFea_33( Sub, nMB );
    % 连接各个分块的LBP直方图形成复合特征向量
    histLBP = zeros(1, 0);
    for ii = 1:r
      for jj = 1:c
        histLBP = [histLBP hist{ii}{jj}];
      end
    end
```

默认情况下函数getLBPHist( )提取(8,2)圆形邻域的LBP特征。如果需要提取3×3的LBP特征，请将代码中getMBLBPFea的调用替换为getMBLBPFea_33的调用。

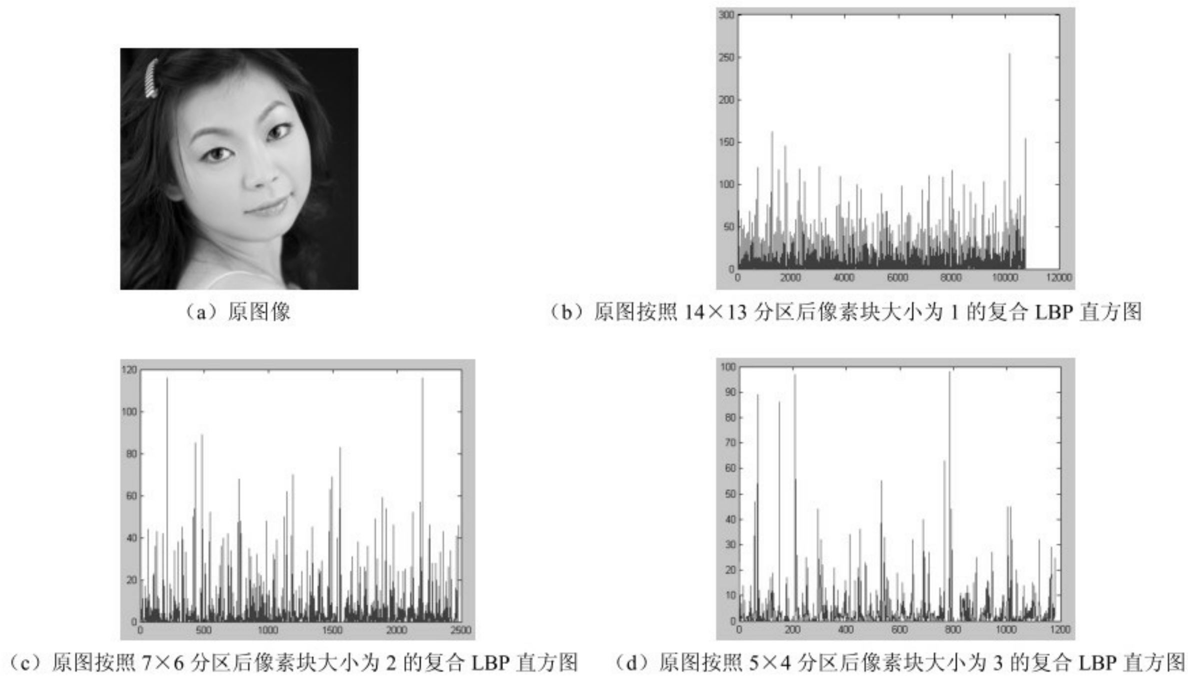
[例12.5]获得一幅图像的复合LBP直方图特征。

以不同的分区数目和不同的块大小从图12.22（a）中提取分区LBP特征的程序如下。

```
>> I = imread('../mh_gray.bmp'); %读入图像
>> histLBP1 = getLBPHist(I, 14, 13, 1); %按照14×13分区后像素块大小为1的复合LBP直方图特征
>> histLBP2 = getLBPHist(I, 7, 6, 2); %按照7×6分区后像素块大小为2的复合LBP直方图特征
>> histLBP3 = getLBPHist(I, 5, 4, 3); %按照5×4分区后像素块大小为3的复合LBP直方图特征
>> figure, plot(histLBP1) %得到图12.22（b）
>> figure, plot(histLBP2) %得到图12.22（c）
```

\>> figure, plot(histLBP3) %得到图12.22（d）

上述程序运行结果如图12.22（b）、（c）、（d）所示。



（a）原图像

（b）原图按照 14×13 分区后像素块大小为 1 的复合 LBP 直方图

（c）原图按照 7×6 分区后像素块大小为 2 的复合 LBP 直方图

（d）原图按照 5×4 分区后像素块大小为 3 的复合 LBP 直方图

▲图12.22 复合LBP直方图特征

图12.22（b）、（c）、（d）分别为对（a）中图像的复合LBP直方图。注意为保证每个分区中的像素块个数基本相同（约为18×21），对于小像素块（$MB_1-LBP_{8,2}^{u2}$），采用较大的分区数目，如（b）中分区数为14×13，这样每个分区中约包含19×20个像素块；而对于较大的像素块（$MB_3-LBP_{8,2}^{u2}$），选择较小的分区数目，如（d）中分区数为 5×4，这样每个分区中约包含 17×21 个像素块。由于每个分区的uniform LBP直方图(向量)维数均为59，（a）中得到的是一个14×13×59 = 10 738维的直方图（向量），（c）中为一个5×4×59 = 1 180维的直方图（向量）。这也说明了越小越精细的分区（分区数目较多）在提供更好的局部描述能力的同时会产生更高维数的复合特征。

对于最终得到的这个复合LBP直方图，既可以将它直接作为1个代

表图像的特征向量来使用，也可以采用12.2.2小节中介绍的方法进一步提取其统计特征。

第13章 图像识别初步

前面章节中介绍的众多的图像处理技术主要就是为图像识别服务的，而从第13章开始将正式转入图像识别领域。通过本章的学习，读者可以建立起对图像识别乃至一般的模式识别问题的基本认识，学习和掌握解决识别问题的一般思路。此外，13.3节还将介绍最小距离分类和模板匹配的简单技术。可以说从现在开始，本书对机器视觉的探究将渐入高潮。

本章的知识和技术热点

- 模式与模式识别的基本概念
- 过度拟合
- 最小距离分类器
- 基于相关的模板匹配

本章的典型案例分析

- 基于最小距离分类器的鸢尾属植物分类
- 基于相关技术的图像模式匹配

13.1 模式识别概述

模式识别（Pattern Recognition）是人类的一项基本智能，在日常生活中，人们经常在进行“模式识别”。随着20世纪40年代计算机的出现以及20世纪50年代人工智能的兴起，人们当然也希望能用计算机来代替或扩展人类的部分脑力劳动。（计算机）模式识别在20世纪60年代初迅速发展并成为一门新学科。

13.1.1 模式与模式识别

模式是由确定的和随机的成分组成的物体、过程和事件。在一个模式识别问题中，它是我们识别的对象。

模式识别是指对表征事物或现象的各种形式的（数值的、文字的和逻辑关系的）信息进行处理和分析，以对事物或现象进行描述、辨认、分类和解释的过程，简单地说就是应用计算机对一组事件或过程进行鉴别和分类。

我们所指的模式识别主要是对语音波形、地震波、心电图、脑电图、图片、照片、文字、符号、生物的传感器等对象进行测量的具体模式进行分类和辨识。

模式识别与统计学、心理学、语言学、计算机科学、生物学、控制论等都有关系。它与人工智能、图像处理的研究有交叉关系。例如自适应或自组织的模式识别系统包含了人工智能的学习机制；人工智能研究的景物理解、自然语言理解也包含模式识别问题。又如模式识别中的预处理和特征提取环节需要应用图像处理的技术；而图像处理中的图像分析也常常应用模式识别的技术。

13.1.2 图像识别

将模式识别的方法和技术应用于图像领域，即当识别的对象是图像时就称为图像识别。虽然对我们人类而言，理解和识别所看见的东西似乎是一件再平常不过的事情，让计算机具有类似的智能却是一项极具挑战性的任务，然而两者在很多环节上是相似的，下面让我们从熟悉的人类视觉过程开始，认识机器的图像识别机理。

图形刺激作用于感觉器官，人们辨认出它是经验过的某一图形的过程,也叫图像再认。所以说在图像识别中,既要有当时进入感官的信息,也要有记忆中存储的信息。只有通过存储的信息与当前的信息进行比较的加工过程，才能实现对图像的再认。这一点和计算机的识别中，需要先

学习一些已经类别的样本（训练样本），才能识别那些类别未知的新样本（测试样本）是相似的。

人的图像识别能力是很强的。图像距离的改变或图像在感觉器官上作用位置的改变，都会造成图像在视网膜上的大小和形状的改变，即使在这种情况下，人们仍然可以认出他们过去知觉过的图像；此外，人类还具有非凡的3D重建能力，比如说您可能只见过某人的正面照片，但您可以认出此人的侧脸甚至是背脸。从这个意义上说，目前计算机的识别能力与我们人类就相差甚远了。

图像识别可能是以图像的主要特征为基础的。每个图像都有它的特征,如字母A有个尖，P有个圈、而Y的中心有个锐角等。相关研究表明，识别时视线总是集中在图像的主要特征上，也就是集中在图像轮廓曲度最大或轮廓方向突然改变的地方，这些地方的信息量最大。而且眼睛的扫描路线也总是依次从一个特征转到另一个特征上。由此可见，在图像识别过程中，知觉机制必须排除输入的多余信息,抽出关键的信息。同时,在大脑里必定有一个负责整合信息的机制，它能把分阶段获得的信息整理成一个完整的知觉映象。这一点正好说明了图像识别中特征提取的必要性。

图像识别中著名的的模板匹配模型认为，要识别某个图像，必须在过去的经验中有这个图像的记忆模式，又叫模板。当前的刺激如果能与大脑中的模板相匹配，这个图像就被识别了。例如有一个字母A,如果在脑中有个A模板,字母A的大小、方位、形状都与这个A模板完全一致,字母A就被识别了。但这种模型强调图像必须与脑中的模板完全匹配才能成功识别，而事实上人不仅能识别与脑中的模板完全一致的图像，也能识别与模板不完全一致的图像。例如,人们不仅能识别某一个具体的字母A,也能识别印刷体的、手写体的、方向不正、大小不同的各种字母A。这就提示我们匹配过程不应基于完全相同的比较而是应基于某种相似性的度量。

13.1.3 关键概念

下面介绍一些识别中常用的重要概念。

● 模式类（pattern class）

模式类是指共享一组共同属性（或特征）的模式集合，通常具有相同的来源。

● 特征（feature）

特征是一种模式区别于另一种模式的相应（本质）特点或特性，是通过测量和或处理能够抽取的数据。

● 噪声（noise）

噪声是指与模式处理（特征抽取中的误差）和（或）训练样本联合的失真，它对系统的分类能力（如识别）产生影响。

● 分类/识别（classification/Recognition）

分类/识别（classification）是指根据特征将模式分配给不同的模式类，识别出模式的类别的过程。

● 分类器（classifier）

分类器可理解为为了实现分类而建立起来某种计算模型，它以模式特征为输入，输出该模式所属的类别信息。

● 训练样本（training sample）

训练样本是一些类别信息已知的样本，通常使用它们来训练分类器。

● 训练集合（training set）

训练样本所组成的集合。

● 训练/学习（training/learning）

训练/学习是指根据训练样本集合，“教授”识别系统如何将输入矢量映射为输出矢量的过程。

● 测试样本（testing sample）

测试样本是一些类别信息对于分类器未知（不提供给分类器其类别信息）的样本，通常使用它们来测试分类器的性能。

● 测试集合（testing set）

测试样本所组成的集合。当测试集合与训练集合没有交集时，称为独立的测试集。

● 测试（testing）

测试是将测试样本作为输入送入已训练好的分类器，得到分类结果并对分类正确率进行统计的过程。

● 识别率（accuracy）

识别率是指对于某一样本集合而言，经分类器识别正确的样本占总样本数的比例。

● 泛化精度（generalization accuracy）

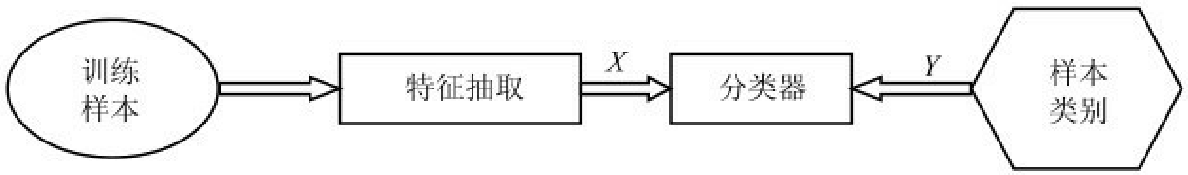
分类器在独立于训练样本的测试集合上的识别率。

13.1.4 识别问题的一般描述

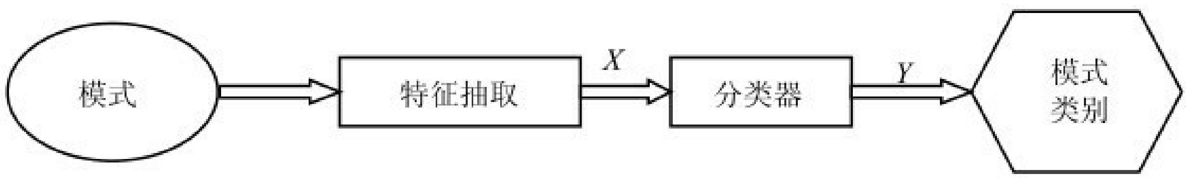
一个模式识别问题一般可描述为：在训练样本集合已经“教授”给识别系统如何将输入矢量映射为输出矢量的前提下，已知一个从样本模式中抽取的输入特征集合（或输入矢量）：$X=\{x_1,x_2,\ldots, x_n\}$，寻找一个根据预定义标准与输入特征匹配的相应特性集合（输出矢量）：$Y=\{y_1,y_2,\ldots,y_m\}$。

这其中对于类别已知的样本参与的训练过程，可参考图13.1（a），此时样本的类别信息Y是已知的，它同训练样本X一起参与分类器的训练；而图13.1（b）中的识别正是利用训练得到的分类器将输入模式X映射为输出类别信息Y的过程。实际上，我们不妨将训练过程理解为一种在输入X和输出Y均已知的情况下确定函数$Y = f(X)$具体形式的函数拟合

过程；而识别过程则可理解为将类别未知的模式X作为f的输入，从而计算出Y的函数求值过程。当然，这里的函数f很可能不具有解析形式，有时会相当复杂，它代表着一种广义上的映射关系。



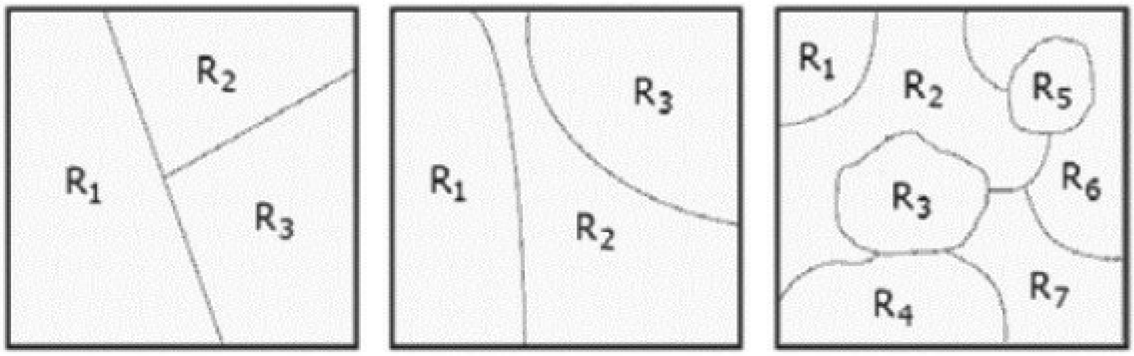
（a）训练过程



（b）识别过程

▲图13.1 训练和识别过程

在第12.1.3小节中讨论特征向量及其几何解释时，我们曾指出了识别（分类）的任务就是找到对特征空间的一种合理划分。分类器将特征空间分成标记为类别的决策区域，对于唯一的分类结果，这些区域必须覆盖整个特征空间且不相交，而每个区域的边缘称为决策边界。从这个意义上说分类器就是分割决策区域的决策边界函数集合，图 13.2 给出了一些典型的决策区域和决策边界。对特征矢量的分类就是确定它属于那个决策区域的过程。



（a）线性的决策边界　（b）二次决策边界　（c）更为复杂的决策边界

▲图13.2 二维空间中的决策区域

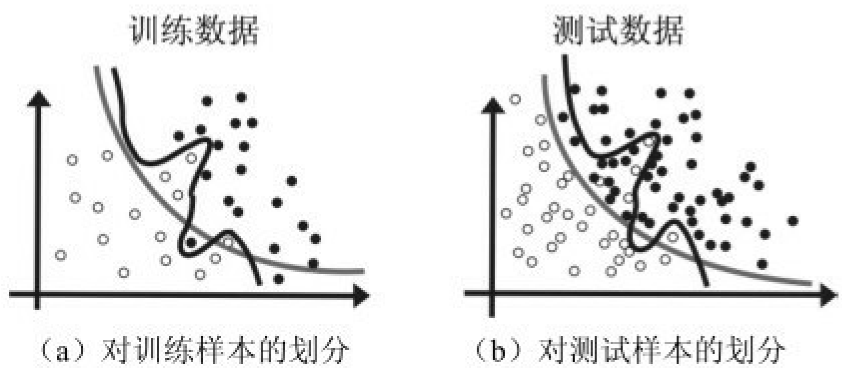
13.1.5 过度拟合

在图13.2中，我们注意到决策边界既可以是图13.2（a）、（b）中那样简单的线性或二次形式，也可以像图 13.2（c）中的那样极其复杂且不规则。那么，对于一个特定的分类问题，我们是应当选择简单的模型还是比较复杂的模型呢？一般来说，简单的模型具有计算复杂度上的优势，训练它们所需的样本数目通常也更少，但它们对空间的划分往往不够精确，导致识别精度受到一定的限制；而复杂的模型可以更好地拟合训练样本，产生非常适应训练数据的复杂决策边界，从而有理由期望它们在测试集上也会有好的表现。然而，这一美好的愿望并不总能实现，事实上，过度复杂的决策边界常常导致所谓“过度拟合”现象的发生，正如例13.1中所描述的那样。

[例13.1]过度拟合现象。

对于图 13.3 中（a）中的两类训练样本，有如图 13.3（a）中所示的两种分类策略，一个简单的二次曲线和另一个复杂得多的不规则曲线。我们看到在图 13.3（a）中不规则曲线完美地分类了所有的训练样本，无一差错；而当面对从未见过的测试样本时[见图 13.3（b）]，复杂曲线的表现令人大失所望，它将一大部分“实心圆”类样本错分为“空心

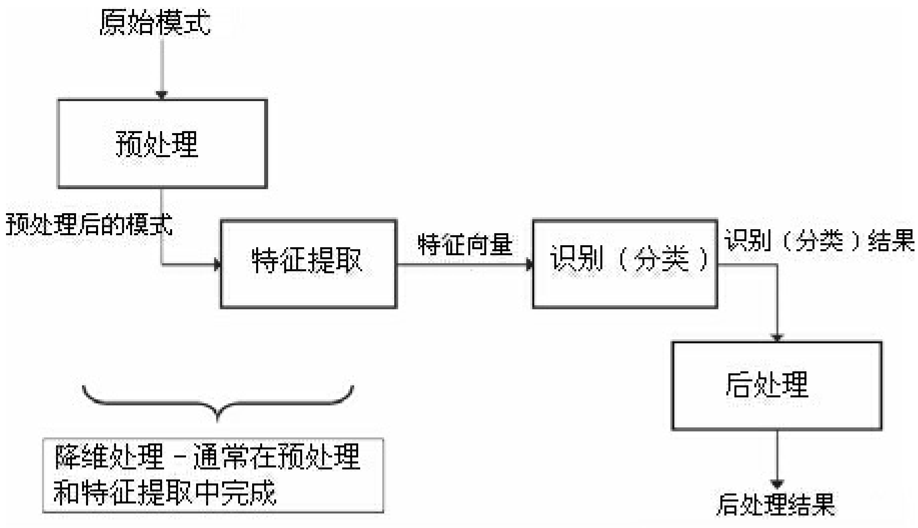
圆”类，简单的二次曲线却工作得相当好。究其原因，主要是过度复杂的决策边界不能够对新数据进行很好地归纳（泛化，一般化），它们过于倾向对训练数据的正确划分（复杂的形式正好为它们完美地拟合训练数据创造了条件），而不能够对真正的数据模型进行很好地分类。这个问题称为过度拟合（Overfit）。简单的决策边界对训练数据不够理想，但是对新数据却往往能够较好地归纳。



▲图13.3 过度拟合

13.1.6 模式识别系统结构

本节最后，图 13.4 为我们展示了一个典型的模式识别系统的结构。原始模式首先经过预处理（本书的第3～11章讨论的主要是图像预处理的方法）；而后经过特征提取（第12章）得到适合分类器处理的特征向量，此过程中有时也包括必要的降维处理；最后分类器输出的识别结果常常还需要后处理，所谓后处理主要指根据得到的识别结果进行评估和改进，像如何调整分类器参数以防止过度拟合等。



▲图13.4 一个典型的模式识别系统

13.1.7 训练/学习方法分类

一般的训练/学习过程是指在给定一般的模型或分类器形式的情况下，利用训练样本去学习和估计模型的未知参数，具体地说就是用某种算法来降低训练样本的分类误差。例如在第14章人工神经网络中将要学习的梯度下降算法，它通过调节分类器的参数，使训练朝着能够降低误差的方向进行。还有很多其他形式的学习算法，通常可分为以下几种形式。

● 教师指导的学习：又称为有监督学习。是指在训练样本集中的每个输入样本类别均已知的情况下进行学习，也就是使用训练模式和相应的类别标记一起来教授分类器。日常生活中有监督学习的一个例子是教孩子识字，教师将字本身（样本）和具体是什么字（类别）一起教给孩子。

● 无教师指导的学习：又称为无监督学习。是指在样本中没有相应的类别信息的情况下，系统对输入样本自动形成“自然的”组织或簇（cluster）。如“聚类算法”就是一种典型的无监督学习。

● 加强学习：又称为基于评价的学习。在加强学习中，并不把类别信息直接提供给分类器，而是让分类器自己根据输入样本计算输出类别，将它与已知的类别标记进行比较，判断对已知训练模式的分类是否正确，从而辅助分类器的学习。日常生活中加强学习的一个例子是提供正确答案的考试讲评，这里考生就相当于分类器，他们先是独立考试（分类），而后根据教师提供的标准答案来改善知识体系（分类器模型）。

13.2 模式识别方法分类

有两种基本的模式识别方法，即统计模式识别（statistical pattern recognition）方法和句法（结构）模式识别（syntactic pattern recognition）方法。统计模式识别是对模式的统计分类方法，即结合统计概率论的贝叶斯决策系统[2]进行模式识别的技术，又称为决策理论识别方法；而利用模式与子模式分层结构的树状信息所完成的模式识别工作，就是句法（结构）模式识别。

13.2.1 统计模式识别

统计模式识别的基本原理是有相似性的样本在模式空间中互相接近，并形成“集团”，即“物以类聚”。其分析方法是根据模式所测得的特征向量$X=(x_{i1},x_{i2},\ldots,x_{id})^T$，（i=1,2,…,N），将一个给定的模式归入C个类$W_1, W_1,\ldots, W_c$中，可视为根据模式之间的某种距离函数来判别分类。其中，T表示转置；N为样本数目；d为样本特征向量的维数。

在统计模式识别中，贝叶斯决策规则从理论上解决了最优分类器的设计问题，但其实施却必须先解决更困难的概率密度估计问题；BP神经网络直接从观测数据（训练样本）学习，是更简便有效的方法，因而

获得了广泛的应用，但它是一种启发式技术，缺乏工程实践的坚实理论基础；统计推断理论研究所取得的突破性成果导致现代统计学习理论——VC维的建立，该理论不仅在严格的数学基础上圆满地回答了人工神经网络中出现的理论问题，而且导出了一种新的学习方法——支持向量机。

13.2.2 句法模式识别

句法模式识别又称结构方法或语言学方法。其基本思想是把一个模式描述为较简单的子模式的组合，子模式又可描述为更简单的子模式的组合，最终得到一个树形的结构描述，在底层的最简单的子模式称为模式基元。

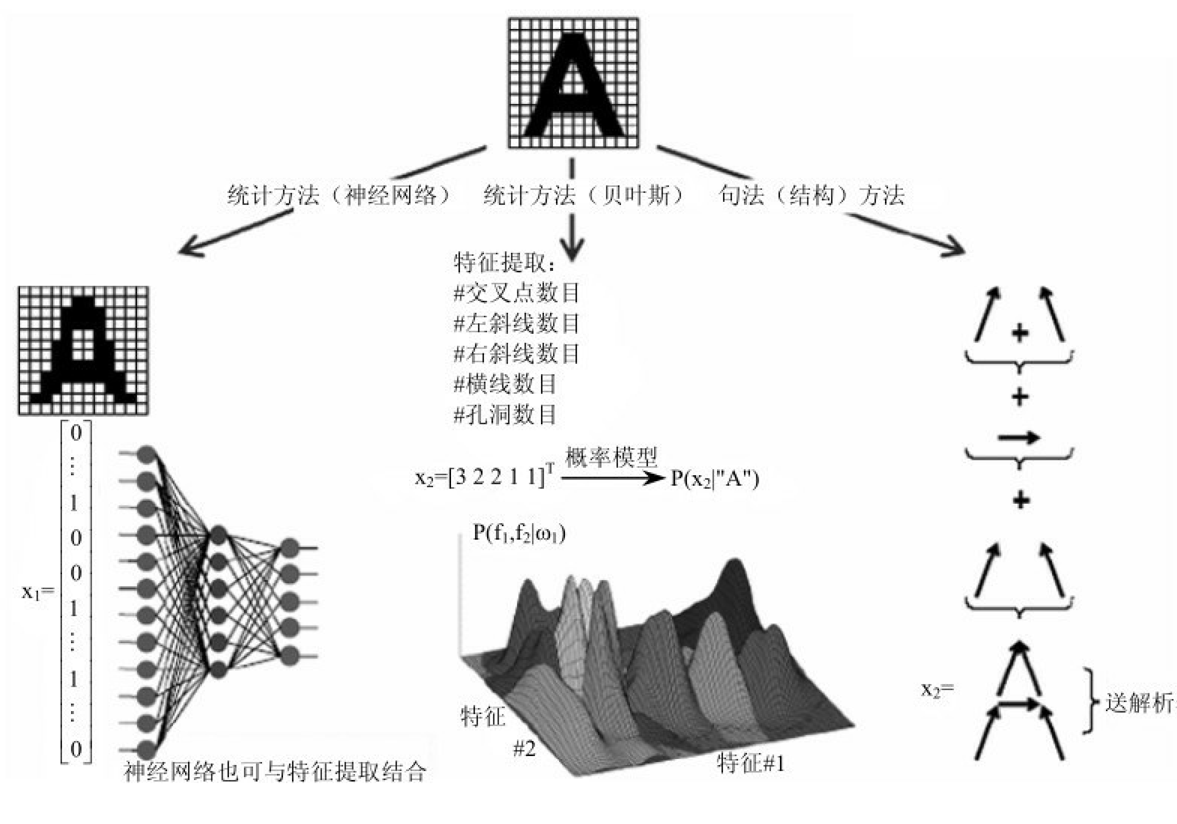
在句法方法中选取基元的问题相当于在统计方法中选取特征的问题。通常要求所选的基元能对模式提供一个紧凑的反映其结构关系的描述，又要易于用非句法方法加以抽取。显然，基元本身不应该含有重要的结构信息。模式以一组基元和它们的组合关系来描述，称为模式描述语句，这相当于在语言中，句子和短语用词组合，词用字符组合一样。基元组合成模式的规则，由所谓语法来指定。一旦基元被鉴别，识别过程可通过句法分析进行，即分析给定的模式语句是否符合指定的语法，满足某类语法的即被分入该类。可以说句法模式识别是基于对结构相似性的测量量来分类模式。该方法不但可以用于分类，也可以用于描述。

[例13.2]统计方法与句法（结构）方法的比较。

图 13.5 给出了对于光学字符识别（OCR）问题统计方法与结构化方法在解决问题上的不同思路。同样对于上方的字母“A”，采用统计方法的一种可能做法是：选取特征为字母中交叉点的数目，左、右斜线的数目，横的数目以及孔洞的数目，这样字母“A”变成了图中所示的1个特征向量$x_2=(3\ 2\ 2\ 1\ 1)^T$，如概率分布已知，则可计算出似然函数

$P(x_2|$“A”$)$，从而构造贝叶斯分类器，这一过程反映在图 13.5 的中间分支上。另一种选择是不经过特征提取，直接将包含字母“A”的矩形区域所有像素的像素值（这里为0或1，白色背景为0，黑色字母为1）按行或按列存储作为特征向量，送给训练好的神经网络进行识别，这一过程如图中左侧分支所示。而当我们采用句法（结构）方法时，字母“A”被看成是图13.5右侧分支所示的一些子结构（比划）的组合，这些子结构有各自的结构和方向并且按照一定的规则组合在一起，最终这些子结构连同它们之间的规则被一并送解析器（Parser）进行类似于句法分析的处理从而识别出类别。

需要说明的是作为一个例子，图 13.5 左侧分支中向量在送神经网络之前没有经过特征提取，这只是选择之一，在实际应用中神经网络的输入也常常是经过特征提取的向量，同样类似于x_2那样经过特征提取的向量可以用作多种分类器的输入，而不仅仅是贝叶斯模型。



▲图13.5 统计方法与句法（结构）方法比较示意

13.2.3 小结

模式识别方法的选择取决于问题的性质。如果被识别的对象极为复杂，而且包含丰富的结构信息，一般采用句法方法；被识别对象不很复杂或不含明显的结构信息，一般采用统计方法。这两种方法不能截然分开，在句法方法中，基元本身就是用统计方法抽取的。在应用中，将这两种方法结合起来分别施加于不同的层次，常能收到较好的效果。

本书并不是一本专门介绍模式识别的书籍，后续的讨论将不涉及句法模式识别的相关内容，这主要是出于对本书内容完整性和紧凑性的考虑（句法模式以自然语言与自动机为其理论根基）；同时我们也不准备从经典的贝叶斯分类理论开始，对各种统计模式识别技术一一讨论，而是将着眼于目前统计模式识别领域中十分活跃、和图像识别关系密切并且已在工程技术领域获得广泛应用的2种非常实用的分类器技术——人工神经网络（第14章）和支持向量机（第15章）,以及广泛应用于特征选择和特征加权的Adaboost技术。

13.3 最小距离分类器和模板匹配

通过前面两节的学习，相信读者已经对模式识别和图像识别有了一定的认识，对其整体框架也有了大致的了解。本节将讨论一种具体的分类技术——最小距离分类器，并附带介绍一种专门针对于图像内容匹配的基于相关的模板匹配技术。

13.3.1 最小距离分类器及其Matlab实现

1. 理论基础

最小距离分类又称最近邻分类，是一种非常简单的分类思想。这种基于匹配的分类技术通过以一种原型模式向量代表每一个类别，识别时一个未知模式被赋予一个按照预先定义的相似性度量与其距离最近的类别，常用的距离度量有欧氏距离、马氏距离等。下面我们以欧氏距离为例讲解最小距离分类器。

一种简单的做法是把每个类所有样本的平均向量作为代表该类的原型，则第i类样本的代表向量为

$$m_i = \frac{1}{N_i}\sum_{x \in w_i} x_i \quad i = 1, 2, \cdots, W \tag{13-1}$$

式中：n_i为第i样本的数目；W_i表示第i类样本的集合；总类别数为W。

当需要对一个未知模式x进行分类时，只需分别计算x与各个m_i（i=1,2,…,W）的距离，然后将它分配给距离最近的代表向量所代表的类别。

对于欧氏距离表示的x与各个m_i的距离，有

$$D_i(x) = \| x - m_i \| \quad i = 1, 2, \cdots, W \tag{13-2}$$

其中，$\|x-m_i\|=((x-m_i)^T (x-m_i))^{1/2}$，表示欧几里的范数，即向量的模。我们在W个$D_i(x)$中找到最小的一个，不妨设为$D_i(x)$，则 x属于第 j类。下面让我们来看一个使用最小距离分类器的实例，请读者思考最小距离分类器具有怎样的决策边界。

2. Matlab实现

下面通过例13.3给出最小距离分类器的实现方法。

[例13.3] 基于最小距离分类器的鸠尾属植物分类

这里仍以Matlab自带的鸢尾属植物数据集为例，要利用最小距离分

类器区分测试集中的样本属于哪一类植物。数据集共有setosa、versicolor和virginica 三类鸢尾属植物。载入fisheriris数据集之后，meas中共包含了150个植物样本，meas矩阵的每一行是代表一个植物样本的特征向量，该特征向量的含义可参见例10.1；而细胞数组species中包含了对应着150个样本的类别信息，从中可见前50个样本为第1类，中间50个为第2类，后50个属于第3类。实验中我们利用每类的前40个样本生成代表该类的模板，后10个被保留至一个独立的测试集，用于验证最小距离分类器的识别率。

计算每个测试样本与3类模板向量最小距离，并且将与测试样本距离最近的模板向量所代表的类作为测试样本的类标号的关键代码如下。

```
for ii = 1:size(Test, 1)
  d(1) = norm(Test(ii, :) - m1); %与第1类的距离
  d(2) = norm(Test(ii, :) - m2); %与第2类的距离
  d(3) = norm(Test(ii, :) - m3); %与第3类的距离
  [minVal class(ii)] = min(d); %计算最小距离并将距离样本最短的类赋给类标签数组 class
end
```

利用最小距离分类器分类3种鸢尾属植物的完整实现代码如下，它位于金羽图书论坛（http://bbs.book95.com）的“金羽图书与答疑”板块与本书同名的主题帖子附件第 13 章目录下的nearest.m文件中。

```
% 例13.3 利用最小距离分类器分类3种鸢尾属植物
load fisheriris %载入Matlab自带的鸢尾属植物数据集
% 每类的前40个样本用于生成代表该类的模板，后10个作为独立的测试样本
m1 = mean( meas(1:40, :) ); %第1类的前40个样本的平均向量
m2 = mean( meas(51:90, :) ); %第2类的前40个样本的平均向量
m3 = mean( meas(101:140, :) ); %第3类的前40个样本的平均向量
```

```
% 测试样本集
Test = [meas(41:50, :); meas(91:100, :); meas(141:150, :)];
% 测试样本集对应的类别标签
classLabel(1:10) = 1;
classLabel(11:20) = 2;
classLabel(21:30) = 3;
% 利用最小距离分类器分类测试样本
class = zeros(1, 30); %类标签
for ii = 1:size(Test, 1)
  d(1) = norm(Test(ii, :) - m1); %与第1类的距离
  d(2) = norm(Test(ii, :) - m2); %与第2类的距离
  d(3) = norm(Test(ii, :) - m3); %与第3类的距离
  [minVal class(ii)] = min(d); %计算最小距离并将距离样本最短的类赋给类标签数组 class
end
% 测试最小距离分类器的识别率
nErr = sum(class ～= classLabel);
rate = 1 - nErr / length(class);
strOut = ['识别率为', num2str(rate)]
```

运行上述程序，最终得到对于30个测试样本的识别率为96.6667%。我们看到，在这个简单的3类问题上，最小距离分类器取得了不错的效果。

13.3.2 基于相关的模板匹配

1．理论基础

基于相关的模板匹配技术可直接用于在一幅图像中寻找某种子图像

模式。回顾在第5章空间域图像增强中，曾经介绍了图像相关的基本概念，对于大小为M×N的图像f(x, y)和大小为J×K的子图像模式w(x, y)，f与w的相关可表示为

$$c(x,y)=\sum_{s=0}^{K}\sum_{t=0}^{J}w(s,t)f(x+s,y+t) \tag{13-3}$$

其中，x = 0, 1, 2, …,N-K, y = 0, 1, 2, …,M-J。这一计算形式与式（5-2）稍有不同，此处的目的是寻找匹配而不是对f(x, y)进行滤波操作，因此w的原点被设置在子图像的左上角，并且式（13-3）给出的形式也完全适用于J和K为偶数的情况。

计算相关c(x, y)的过程就是在图像f(x，y)中逐点地移动子图像w(x, y)，使w的原点和点(x，y)重合，然后计算w与f中被w覆盖的图像区域对应像素的乘积之和，以此计算结果作为相关图像c(x, y)在(x, y)点的响应。

相关可用于在图像f(x, y)中找到与子图像w(x, y)匹配的所有位置。实际上，当w按照上一段中描述的过程移过整幅图像f之后，最大的响应点(x_0, y_0)即为最佳匹配的左上角点。我们也可以设定一个阈值T，认为响应值大于该阈值的点均是可能的匹配位置。

相关的计算是通过将图像元素和子模式图像元素联系起来获得的，将相关元素相乘后累加。我们完全可以将子图像w视为一个按行或按列存储的向量$\vec{b}$，将计算过程中被w覆盖的图像区域视为另一个按照同样的方式存储的向量$\vec{a}$。这样一来，相关计算就成了向量之间的点积运算。

两个向量的点积为

$$\vec{a}\bullet\vec{b}=|\vec{a}||\vec{b}|\cos\theta \tag{13-4}$$

其中，θ为向量$\vec{a}$、$\vec{b}$之间的夹角。显然，当$\vec{a}$和$\vec{b}$具有完全相同的方向（平行）时，cosθ=1，从而式（13-4）取得其最大值$|\vec{a}||\vec{b}|$，这就意味着当图像的局部区域类似于子图像模式时，相关运算产生最大的响应。然而，式（13-4）最终的取值还与向量$\vec{a}$、$\vec{b}$自身的模有关，这将导致按照式（13-4）计算的相关响应存在着对f和w的灰度幅值比较敏感的缺陷。这样一来，在f的高灰度区域，可能尽管其内容与子图像w的内容并不相近，但由于$|\vec{a}|$自身较大而同样产生一个很高的响应。可通过对向量以其模值来归一化从而解决这一问题，即通过$\frac{\vec{a}\bullet\vec{b}}{|\vec{a}||\vec{b}|}$来计算相关。

改进的用于匹配的相关计算公式如下。

$$r(x,y)=\frac{\sum_{s=0}^{K}\sum_{t=0}^{J}w(s,t)f(x+s,y+t)}{\left[\sum_{s=0}^{K}\sum_{t=0}^{J}w^2(s,t)\cdot\sum_{s=0}^{K}\sum_{t=0}^{J}f^2(x+s,y+t)\right]^{1/2}} \tag{13-5}$$

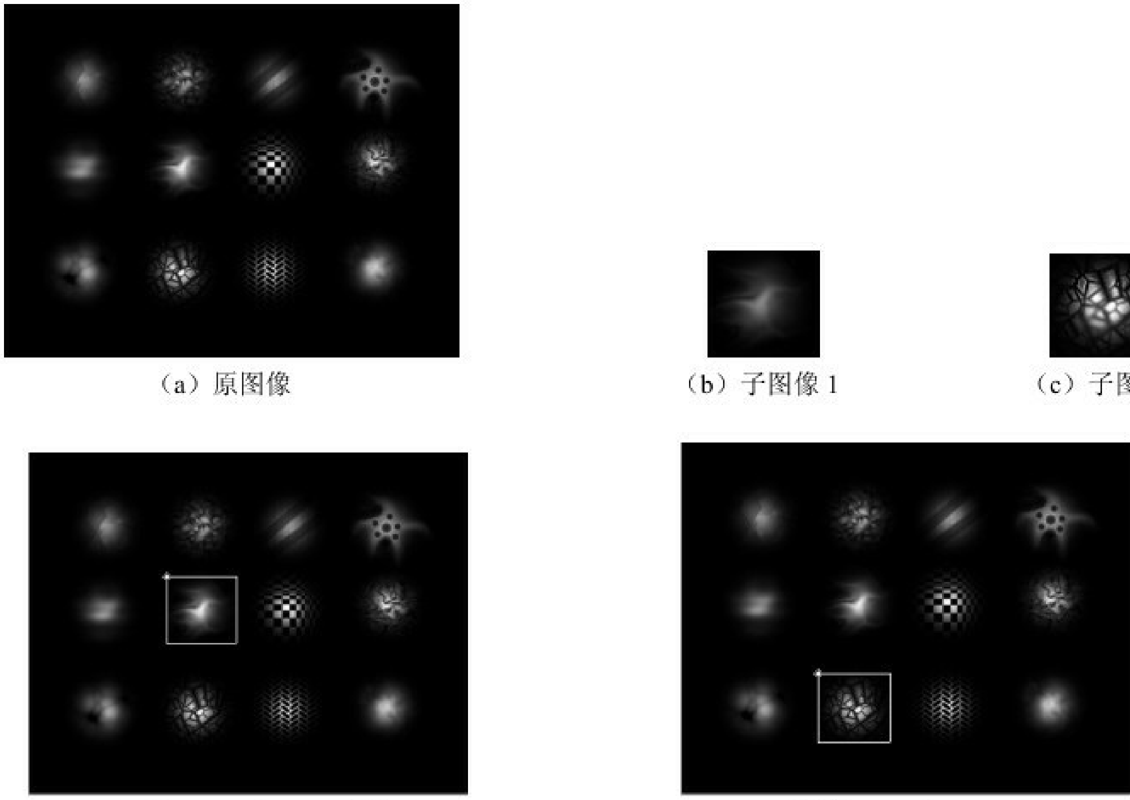
改进的相关公式实际上计算的是向量$\vec{a}$、$\vec{b}$之间的夹角余弦值。显然，它只和图案模式本身的形状或纹理有关，与幅值（亮度）无关。

2．Matlab实现

下面通过例13.4给出基于相关的模板匹配的实现方法。

[例13.4]基于相关的图像匹配。

图13.6（a）中黑色背景的图像中包含有12种不同的图案模式，要在图13.6（a）中找到图13.6 （b）和图13.6（c）中的子图像的最佳匹配。图13.6（b）中的图像对应着图13.6（a）中第2行的第2个小图案，但整体亮度较其在图13.6（a）中更暗，而图13.6（c）中的图像则与图13.6（a）中第3行的第2个小图案相似，略有区别。



（a）原图像　（b）子图像 1　（c）子图像 2

（d）子图像（b）在（a）中的匹配　（e）子图像（c）在（a）中的匹配

▲图13.6 基于相关的图像匹配

我们编写了 imcorr 函数实现图像相关，代码如下。该函数被封装在金羽图书论坛（http://bbs.book95.com）的“金羽图书与答疑”板块与本书同名的主题帖子附件的“chapter13/code/”目录下的imcorr.m文件中。

```
function Icorr = imcorr(I, w)
% function corr = imcorr(I, w, )
% 计算图像 I 与子模式 w 的相关响应，并提示最大的响应位置
%
% Input：I - 原始图像
%        w - 子图像
%
% Output：Icorr - 响应图像
```

```
[m, n] = size(I);
[m0, n0] = size(w);
Icorr = zeros(m-m0+1, n-n0+1); %为响应图像分配空间
vecW = double( w(:) ); %按列存储为向量
normW = norm(vecW); %模式图像对应向量的模
for ii = 1:m-m0+1
  for jj = 1:n-n0+1
    subMat = I(ii:ii+m0-1, jj:jj+n0-1);
    vec = double( subMat(:) ); %按列存储为向量
    Icorr(ii, jj) = vec' * vecW / (norm(vec)*normW+eps); %计算当前位
置的相关
  end
end
% 找到最大响应位置
[iMaxRes, jMaxRes] = find(Icorr == max( Icorr(:) ) );
figure, imshow(I);
hold on
for ii = 1:length(iMaxRes)
  plot(jMaxRes(ii), iMaxRes(ii), '*'); %绘制最大响应点
  % 用矩形框标记出匹配区域
  plot([jMaxRes(ii), jMaxRes(ii)+n0-1], [iMaxRes(ii), iMaxRes(ii)] );
  plot([jMaxRes(ii)+n0-1, jMaxRes(ii)+n0-1], [iMaxRes(ii),
iMaxRes(ii)+m0-1] );
  plot([jMaxRes(ii), jMaxRes(ii)+n0-1], [iMaxRes(ii)+m0-1,
iMaxRes(ii)+m0-1] );
  plot([jMaxRes(ii), jMaxRes(ii)], [iMaxRes(ii), iMaxRes(ii)+m0-1] );
end
```

imcorr函数中根据相关响应的结果，分别将对图13.6（b）和图13.6（c）响应的最大值用“*”号标记在图13.6（d）和图13.6（e），并用矩形给出了相应的匹配区域。该函数的调用方式如下。

```
>> I = imread('patterns.bmp');
>> I1 = imread('pat1.bmp');
>> I2 = imread('pat2.bmp');
>> J1 = imcorr(I, I1);
>> max(J1(:))
ans =
   1.0000
>> J2 = imcorr(I, I2);
>> max(J2(:))
ans =
   0.9784
```

上述程序运行后对图13.6（b）、（c）子图像的匹配结果如图13.6（d）、（e）所示。我们发现，两次匹配都只找到了一个最大值，由于图13.6（a）中包含了与图13.6（b）中子图像纹理结构完全相同的图像区域，只是整体明暗不同，因此相关响应中最大值max(J1(:)) = 1；而与图13.6（c）的相关响应最大值为0.9784，这也正说明了图13.6（c）中子图像模式与图13.6（a）中的最佳匹配区域并不完全相同。

虽然我们之前通过向量模值的归一化得到了幅值（亮度）不变的相关匹配算子，但相关计算仍然对尺寸和旋转变换非常敏感。如果子图像w与图像f中对应的相似目标大小不同，则一般来说很难得到满意的匹配结果。为了解决这一问题，有时需要采用多种分辨率扫描原图像f，计算相关，在各个分辨率下寻找可能的匹配位置，这在计算量上的需求可想而知，因而很难在实际系统中使用。类似的，如果旋转变化的性质是未知的，则寻找最佳匹配就要求对w进行全方位的旋转。更多的时候，

我们需要利用对问题的先验知识得到有关尺寸和旋转变换方式的一些线索，从而可以借助几何变换中的一些技术在匹配之前对这些变换进行归一化处理。

13.3.3 相关匹配的计算效率

一般子图像模式w总是比图像f要小得多。尽管如此，除非w非常小，否则例13.4中采用的空间相关算法的计算量都会比较大，以至于总是要依靠硬件来实现。

一种提升计算效率的方法是在频域中实现相关，回忆第6章中曾学习过的卷积定理，它为我们建立了空间卷积和频域乘积之间的对应关系。类似的，通过相关定理可以将空间相关与频域乘积联系起来。相关定理中指出了两个函数的空间相关可以用一个函数的傅立叶变换同另一个函数的傅立叶变换的傅共轭的乘积的傅立叶逆变换得到，当然反过来也成立，即

$$f(x,y)\circ w(x,y) \Leftrightarrow F(u,v)H^{*}(u,v) \quad （13\text{-}6）$$

$$f(x,y)w^{*}(x,y) \Leftrightarrow F(u,v)\circ H(u,v) \quad （13\text{-}7）$$

其中，“D”表示相关，“*”表示复共轭。

下面给出了按照上述思路在频域下实现相关，再变换回空域得到响应图像Icorr的Matlab实现函数dftcorr，它被封装在金羽图书论坛（http://bbs.book95.com）的“金羽图书与答疑”板块与本书同名的主题帖子附件的“chapter13/code/”目录下的dftcorr.m文件中。程序在原图像I中标记出了匹配位置。

```
function Icorr = dftcorr(I, w)
% function Icorr = dftcorr(I, w)
% 在频域下计算图像 I 与子模式 w 的相关响应，并提示最大响应位
```

置

```
%
% Input：I - 原始图像
%        w - 子图像
%
% Output：Icorr - 响应图像
I = double(I);
[m n] = size(I);
[m0 n0] = size(w);
F = fft2(I);
w = conj(fft2(w, m, n)); %w 频谱的共轭
Ffilt = w .* F; %频域滤波结果
Icorr = real(ifft2(Ffilt)); %反变换回空域
% 找到最响相应位置
[iMaxRes, jMaxRes] = find(Icorr == max( Icorr(:) ) );
figure, imshow(I, []);
hold on
for ii = 1:length(iMaxRes)
  plot(jMaxRes(ii), iMaxRes(ii), 'w*');
  plot([jMaxRes(ii), jMaxRes(ii)+n0-1], [iMaxRes(ii), iMaxRes(ii)], 'w-'
);
  plot([jMaxRes(ii)+n0-1, jMaxRes(ii)+n0-1], [iMaxRes(ii),
iMaxRes(ii)+m0-1], 'w-' );
  plot([jMaxRes(ii), jMaxRes(ii)+n0-1], [iMaxRes(ii)+m0-1,
iMaxRes(ii)+m0-1], 'w-' );
  plot([jMaxRes(ii), jMaxRes(ii)], [iMaxRes(ii), iMaxRes(ii)+m0-1], 'w-
' );
```

end

对于例13.4中的模板匹配问题，和imcorr相比，dftcorr在执行效率上的优势是显而易见的。这是因为该问题中模板图像的大小为 61×64，虽然比原图像小得多，但已可以说是一个比较大的模板了。有文献（Campbell 1969）曾指出，如果w中的非零元素数目小于132（大约13×13见方），则直接在空域中计算相关较为划算，否则通过上述方法变换至频域下计算更为合适。当然，这个数目不是绝对的，它还与f的大小以及运算机器本身有关，读者可以把它作为参考，来决定在应用系统中实现相关的具体方式。

第14章 人工神经网络

在对图像识别有一个整体认识之后，从现在开始我们要学习两种高实用性的分类技术，本章介绍人工神经网络，下一章将围绕支持向量机展开讨论，对它们的理论基础将给出必要的介绍，重点放在两者在应用和实现层面的一些技巧和注意事项，学习的目的是使读者在学习之后立即可以在工程实践中获益。

本章的知识和技术热点

- ANN的基本结构
- 反向传播算法
- ANN的训练和使用技巧

本章的典型案例分析

- 基于ANN的字母模式识别系统

14.1 人工神经网络简介

人工神经网络（Artificial Neural Networks, ANN）也简称为神经网络（NN），是对人脑或生物神经网络（Natural Neural Network）若干基本特性的抽象和模拟。它为从样本中学习值为实数、离散值或向量的函数提供了一种健壮性很强的解决方案，已经在诸如汽车自动驾驶、光学字符识别（OCR）和人脸识别等很多实际问题中取得了惊人的成功。

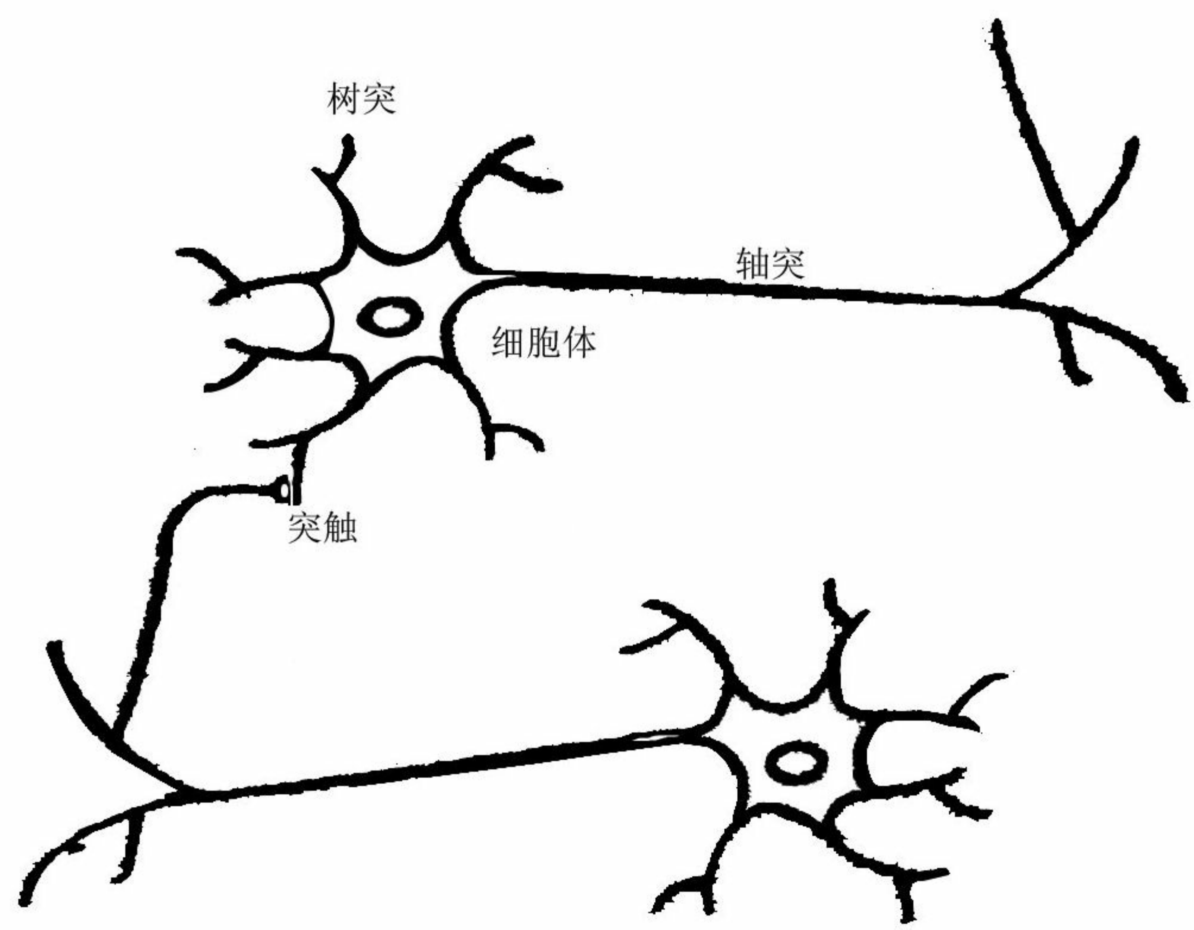
对于一个分类问题，我们的目标就是学习决策函数h(x)，该函数的输出为离散值（类标签）或者向量（经过编码的类标签），ANN自然能够胜任这一任务；此外，由于可学习实值函数，ANN也是函数拟合的利器。本书中对于ANN的介绍只限于分类问题，将涉及输出为离散值和向量的情况。

14.1.1 仿生学动机

1．生物神经网络

众所周知，生物大脑由大量的神经细胞（神经元，neuron）组成，这些神经元相互连接成十分复杂的网络。每个神经元由3部分组成：树突、细胞体和轴突，如图14.1所示。树突是树状的神经纤维接受网络，它将输入的电信号传递给细胞体；轴突是单根长纤维，它把细胞体的输出信号导向其他神经元。大量这样的神经元广泛地连接从而形成了网络。神经元的数目、排列拓扑结构以及突触的连接强度决定了生物神经网络的功能。

神经元之间利用电化学过程传递信号。一个神经元的输入信号来自另一些神经元的输出，这些神经元的轴突末梢与该神经元树突相遇形成突触。大脑神经元有两种状态：兴奋和抑制。细胞体对这些输入信号进行整合并进行阈值处理。如果整合后的刺激值超过某一阈值神经元被激活而进入兴奋状态，此时就会有一个电信号通过轴突传递给其他神经元；否则神经元就处于抑制状态。



▲图14.1 生物神经元及其连接方式

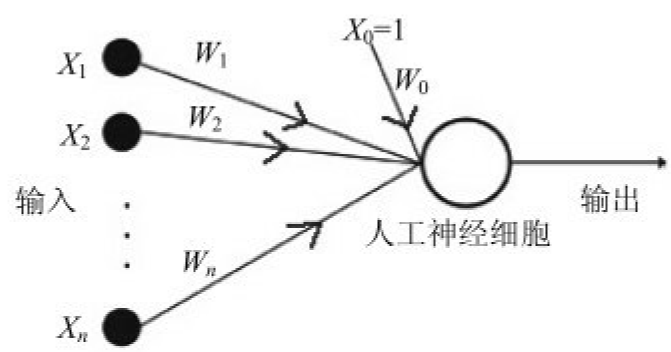
生物大脑具有超强的学习能力。相关研究表明，如果一个神经元在一段时间内频繁受到激励，则它与连接至输入的神经元之间的连接强度就会相应地改变，从而使得该神经元细胞再次受到激励时更易兴奋；相反，一个某段时间内不被激励的神经元的连接有效性会慢慢地衰减。这一现象说明神经元之间的连接具有某种可塑性（可训练性）。

一个训练生物神经网络的例子如人类学习下棋。起初在没有接受过任何下棋训练时，可以理解为大脑神经元网络处于一个随机状态，对于某一个棋局（输入），产生一个随机的应对策略（网络的实际输出）；接下来，这一应对策略会从指导教师（教下棋的人或者是一本棋谱）那里得到相应的反馈（这一步下得好还是不好，或者正确的走法是什么，

相当于训练样本的目标输出），并以此反馈作为调整网络神经元之间连接（这些连接具有可训练性）的依据；随着这种训练和调整过程的进行，网络对于新的棋局（输入）的决策越来越接近于最优决策（实际输出更加接近于目标输出），网络处于一种善于对某个棋局做出正确反映的状态。这就是我们在学习下棋时从一个初学者到高手的过程。

2．人工神经网络

人工神经网络（ANN，以下简称网络或神经网络）的研究在相当程度上受到了生物大脑的仿生学启发，它由一系列简单的人工神经元相互密集连接构成，其中每个神经元同样由3部分组成：输入、人工神经细胞体和输出。每个神经元具有一定数量的实数值输入，并产生一个的实数值的输出，如图14.2所示。



▲图14.2 人工神经元示意图

一个人工神经元的输入信号来自另一些神经元的输出，其输出又可以作为另一些神经元的输入。一种称为感知器（见 14.2.1 小节）的人工神经元同样具有两种状态：1和–1。人工神经细胞对这些输入信号进行整合并进行阈值处理。如果整合后的刺激值超过某一阈值神经元被激活而进入1状态；否则神经元就处于–1状态。

正如大脑可以通过不断调节神经元之间的连接而达到不断学习进步的目的，ANN也可以通过不断调整输入连接上的权值以使得网络更加适

应训练集合。

在ANN的训练过程中，训练样本特征向量是ANN的输入，训练样本的目标输出（在分类问题中为类别信息）是网络的输出。初始情况下，网络权值被初始化为一种随机状态，当把某个训练样本输入网络时，由此产生的网络输出与训练样本目标输出之间的差异称为训练误差；接下来， ANN将根据某种机制（参见14.2.1小节）调节权值w，使得训练误差逐步减小；随着这种训练和调整过程的进行，网络对于训练样本的实际输出将越来越接近于目标输出。

人工神经网络和生物神经网络机能的类比如表14.1所示。

表14.1 人工神经网络与生物神经网络机能类比

人工神经网络（ANN）	生物神经网络
输入	树突
输出	轴突
人工神经细胞	生物神经细胞体
人工神经元的 1、－1 两种状态	生物神经元的兴奋和抑制状态
ANN 的训练	大脑的学习
网络权值的调整	生物神经细胞之间连接的调节
ANN 对于特定输入样本的输出	大脑对于特定输入情况的决策

14.1.2 人工神经网络的应用实例

在准备稍显枯燥的ANN理论学习之前，先来看一个将ANN应用于学习汽车自动驾驶的典型案例，了解ANN能够解决什么样的问题并从中获得足够的对于ANN的感性认识。

[例 14.1] ALVINN汽车自动驾驶系统

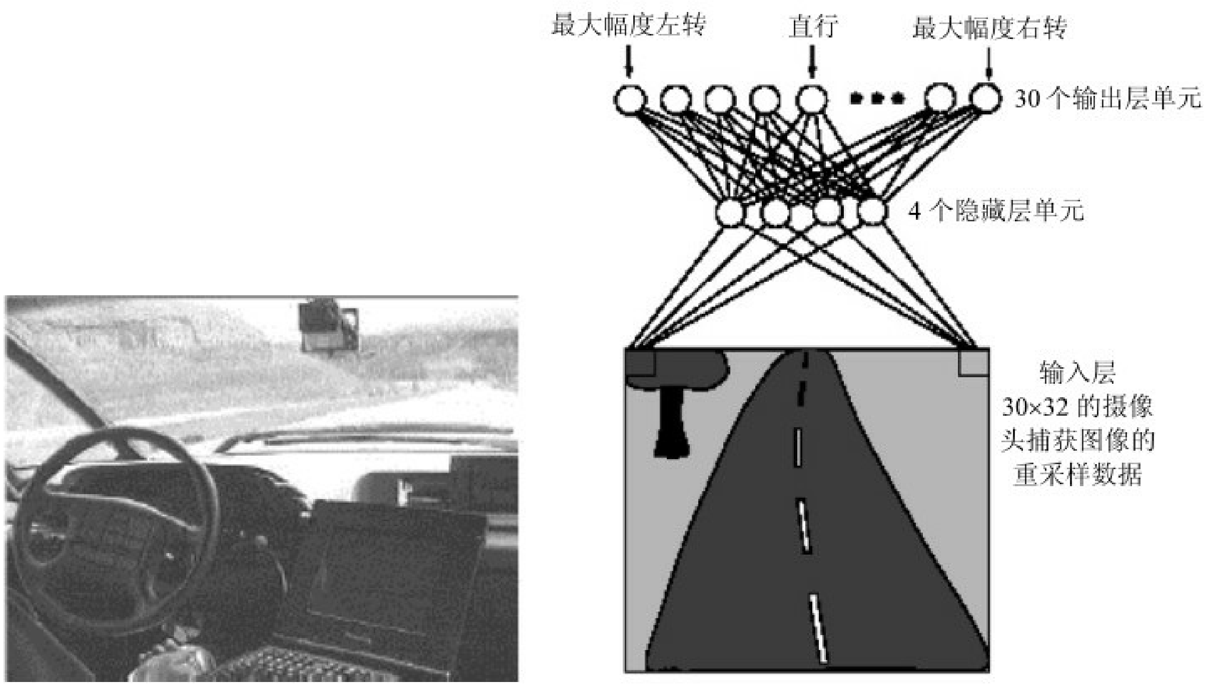
著名的自动驾驶系统ALVINN是ANN一个典型的应用。该系统使用一个经过训练的ANN以正常速度在高速公路上驾驶汽车。如图14.3（b）所示，ALVINN具有一个典型的3层结构，网络的输入层共有

30 × 32个单元，对应于一个30×32像素点阵，是由一个安装在车辆上的前向摄像机获取的图像经过重采样得到的。输出层共有30个单元，输出情况指出了车辆行进的方向。

在训练阶段，ANVINN 以人类驾驶时摄像机所捕获的前方交通状况作为输入，以人类通过操作方向盘给出的前进方向作为目标输出，整个训练过程大约5min；在测试阶段，ALVINN用学习到的网络在高速公路上以70英里的时速成功地驾驶了90英里。

摄像头捕获图像的30×32的重采样图像被作为网络的输入，对应于960个输入层单元，这些输入又连接至4个隐藏单元，再连接到30个输出单元，输出为一个30维向量，相当于把整个方向盘的控制范围分成30份，每个输出单元对应一个特定的驾驶方向，决策结果为输出值最大的单元对应的行驶方向

在ALVINN中所有单元分层互连形成了一个有向无环图，相邻层之间是全连接的，这是很多ANN的典型结构。ANN结构还可有多种其他类型，本章只集中介绍具有类似于ALVINN的网络结构并以反向传播（BP, Back Propogation）算法为基础的最常见和实用的ANN。



▲图14.3 学习汽车自动驾驶的 ALVINN系统

14.2 人工神经网络的理论基础

在进入用于ANN训练的反向传播算法的学习之前，首先了解用于训练线性单元的梯度下降法是非常有益的，它构成了反向传播算法的基础。不同的是梯度下降法只是训练一个线性单元，而反向传播算法能够训练多个单元的互连网络。因此我们将从感知器开始，进而介绍训练线性单元的梯度下降算法开始，最终推广到用于神经网络训练的反响传播算法。

14.2.1 训练线性单元的梯度下降算法

1．感知器（**perceptron**）

感知器是一种只具有两种输出的简单的人工神经元，一个具有n个

实数输入$x_1,x_2,\ldots, x_n$的感知器如图14.4所示。每个输入x_i对应一个权值W_i，此外还有一个偏置（BIAS）项W_0，首先需计算这n个输入根据其权值形成的一个线性组合再加上偏置项W_0，即

$$\text{net} = \sum_{i=1}^{n} w_i \cdot x_i + w_0 \quad （14\text{-}1）$$

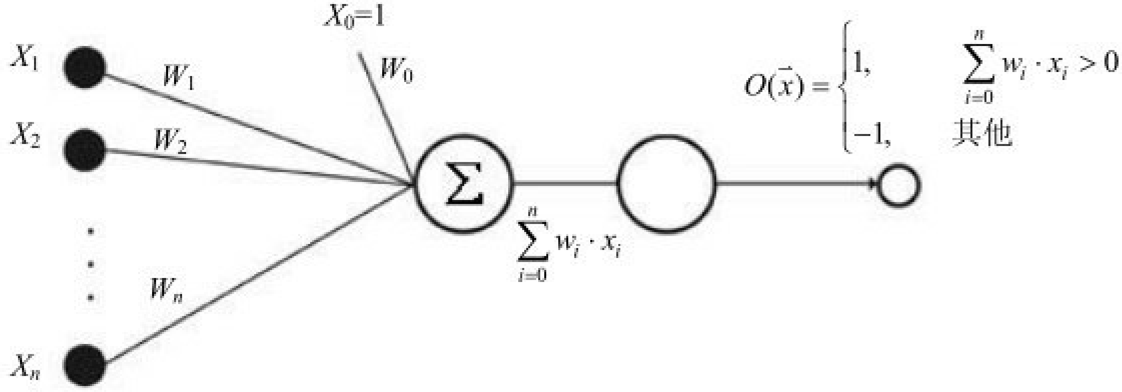
不妨令$x_0=1$，从而式（14-1）可表示为

$$\text{net} = \sum_{i=0}^{n} w_i \cdot x_i = \vec{w} \cdot \vec{x} \quad （14\text{-}2）$$

其中输入向量$\vec{x}=(1,x_1,x_2,\ldots,x_n)$，权向量$\vec{w}=(W_0,W_1,W_2,\ldots,W_n)$。感知器的输出为式（14-2）经过阈值化处理的结果：

$$O(\vec{x}) = \begin{cases} 1, & \sum_{i=0}^{n} w_i x_i > 0 \\ -1, & \text{其他} \end{cases} \quad （14\text{-}3）$$

感知器的工作方式与生物神经单元颇为相似。注意到在图 14.4 中每个输入到Σ求和单元的连接上的权值W_i表示在对输入的线性组合中各个 x_i的贡献大小，Σ单元的加权求和处理即为对输入的整合过程。而感知器的1和–1两种输出对应于生物神经元的兴奋和抑制两种状态。



▲图14.4 感知器单元

由式（14-3）的形式可知，感知器对应于一个n维空间中的超平面 $\vec{w}\cdot\vec{x}$ =0，它能够分类两类样本。对于其一侧的输入样本输出1，对于另一侧的样本输出–1。训练过程就是调整权值$W_1,W_2,\dots,W_n$，使得感知器对于 2类样本分别输出+1和–1。

2．线性单元（**linear unit**）

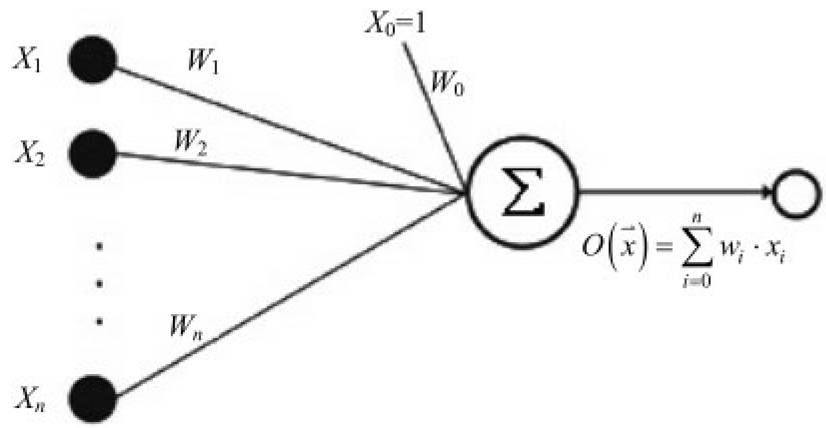
只有1和–1两种输出限制了感知器的处理和分类能力，一种简单的推广是线性单元，即不带阈值的感知器。一个具有n个输入$x_1,x_2,\dots,x_n$的线性单元如图14.5所示，其输出为其n个输入根据其权值形成的一个线性组合再加上偏置项W_0，即

$$O(\vec{x}) = \sum_{i=1}^{n} w_i \cdot x_i + w_0 \qquad （14\text{-}4）$$

令x_0=1，从而式（14-1）可表示为

$$O(\vec{x}) = \sum_{i=0}^{n} w_i \cdot x_i = \vec{w}\cdot\vec{x} \qquad （14\text{-}5）$$

式中：输入向量$\vec{x}$ =$(1,x_1,x_2,\dots,x_n)$；权向量$\vec{w}$ =$(W_0,W_1,W_2,\dots,W_n)$。



▲图14.5 线性单元示意图

训练线性单元的核心任务就是调整权值$W_1,W_2,\ldots,W_n$，使得线性单元对于训练样本的实际输出与训练样本的目标输出尽可能地接近。

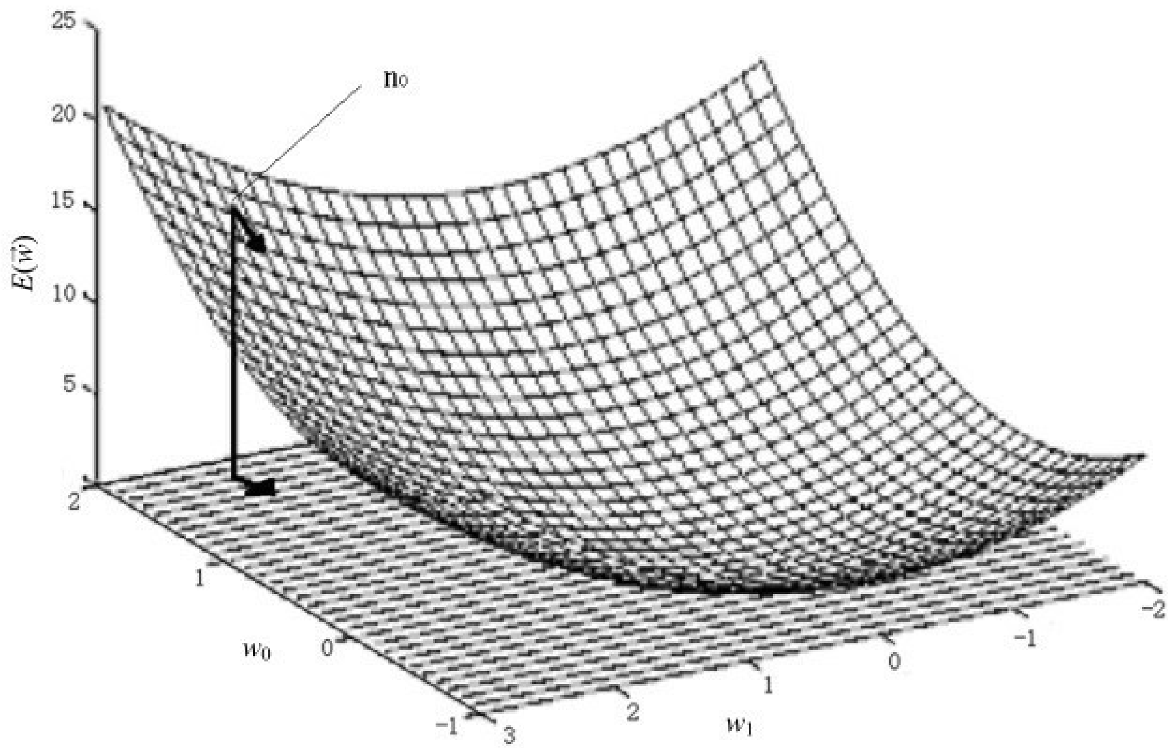
3．误差准则

为了推导线性单元的权值学习法则，首先必须定义一个度量标准来衡量在当前权向量$\vec{w}$下ANN相对于训练样例的训练误差（training error）。一个常见的度量标准为平方误差准则。

$$E(\vec{w})=\frac{1}{2}\sum_{d\in D}(t_d-o_d)^2 \qquad （14-6）$$

式中：D是训练样本集合；t_d是训练样本d的目标输出（训练样本d的类别信息）；o_d是线性单元对于训练样本d的实际输出，即O（x_d）；$E(\vec{w})$是目标输出t_d和实际输出o_d的差的平方在所有训练样本上求和的1/2倍，这里常数1/2主要是为了最终的推导结果在形式上的简洁（在推导过程中与其后平方项求导产生的因子2抵消）。观察式（14-3），对于特定的问题，训练集合就固定了下来，因此$\vec{x}$是定值，目标输出t_d也为定值（训练样本的类别信息是已知的），而o_d只依赖于权向量$\vec{w}$，可以把E写成是权向量$\vec{w}$的函数。

图 14.6 给出了在解空间中搜索的可视化解释。由于无法绘制出超过三维的空间，这里设$\vec{w}$=(W_0,W_1)，即我们要在二维解空间W_0-W_1中搜索最小化式（14-3）的权向量$\vec{w}$，第三维（纵轴）表示误差$E(\vec{w})$，图中给出了在整个二维搜索空间W_0-W_1上的误差曲面$E(\vec{w})$。由于$E(\vec{w})$是关于$\vec{w}$的二次函数，这个误差曲面必然是具有单一的全局最小值的抛物面。当$\vec{w}$是更高维的权向量时，只不过相当于在一个更高维的解空间中搜索最小化$E(\vec{w})$的解$\vec{w}^*$，误差曲面$E(\vec{w})$也就成了一个超抛物面。



▲图14.6 误差曲面

为了确定一个使E最小化的权向量，我们从任意的初始权向量$\vec{w}^0$开始，然后以很小的步长反复修改这个权向量，而每一步的修改都能够使误差E减小。继续这个过程直到找到全局的最小值点$\vec{w}^*$。

很自然我们希望上述过程越快越好，因此在每一步都要使 E 减小尽

可能快，这样的寻优过程同样可在图14.6中找到可视化的解释。设想你从解空间中的任意初始点n_0开始出发，目的地是误差曲面的最低点，由于能见度有限等原因，某一时刻你只能看到距离自己很近的周边区域，因此每次只试探性地跨出一小步，为了以最快速度到达谷底，一个合理的选择是找到目前最陡峭的下降方向，朝该方向跨出这一步，在新的位置上你将获得新的视野，也将找到新的最陡方向，不断重复这一过程直到到达最低目标点。

4．梯度下降法（**gradient descent**）的推导

怎样才能计算出沿误差曲面最快的下降方向呢？我们在高等数学中学习过梯度（gradient）的概念，而梯度是方向导数最大的方向，因此可通过计算E相对于向量$\vec{w}$的每个分量的偏导数来得到方向导数最大的方向——梯度，记作$\nabla E(\vec{w})$。

$$\nabla E(\vec{w}) = [\frac{\partial E}{\partial w_0}, \frac{\partial E}{\partial w_1}, \ldots, \frac{\partial E}{\partial w_n}] \qquad (14\text{-}7)$$

需要注意的是梯度$\nabla E(\vec{w})$本身是一个表示方向导数最大方向的向量，因此它对应于E最快的上升方向，而我们要寻找的沿误差曲面最快的下降方向自然就是负梯度$-\nabla E(\vec{w})$。所以梯度下降的训练法则应为

$$\vec{w} \leftarrow \vec{w} + \Delta\vec{w} \qquad (14\text{-}8)$$

式中：

$$\Delta\vec{w} = -\eta\nabla E(\vec{w}) \qquad (14\text{-}9)$$

这里η是一个称为学习率的正的常数，它决定了梯度下降搜索中的步长。$\vec{w}$代表解空间中的当前搜索点，$\Delta\vec{w}$代表向当前最快下降方向的一小段位移，$\vec{w} \leftarrow \vec{w} + \Delta\vec{w}$则表示在搜索空间中从当前点沿最快下降方

向移动一小段距离并更新当前位置至此移动位置。

训练法则式（14-8）也可以写成它的分量形式。

$$\vec{w}_i \leftarrow \vec{w}_i + \Delta\vec{w}_i \tag{14-10}$$

式中：

$$\Delta\vec{w}_i = -\eta \frac{\partial E}{\partial w_i} \tag{14-11}$$

这样很明显，下降最快的方向可以按照比例∂E/∂W_i改变$\vec{w}$中的每一个W_i来实现。剩下问题的就是计算∂E/∂W_i了。

$$\begin{aligned}\frac{\partial E}{\partial w_i} &= \frac{\partial}{\partial w_i}[\frac{1}{2}\sum_{d\in D}(t_d - o_d)^2] = \frac{1}{2}\sum_{d\in D}\frac{\partial}{\partial w_i}(t_d - o_d)^2 \\ &= \frac{1}{2}\sum_{d\in D}2(t_d - o_d)\frac{\partial}{\partial w_i}(t_d - o_d) \\ &= \sum_{d\in D}(t_d - o_d)\frac{\partial}{\partial w_i}(t_d - \vec{w}\cdot\vec{x}_d) \\ &= \sum_{d\in D}(t_d - o_d)(-x_{id})\end{aligned} \tag{14-12}$$

式中：x_{id}表示训练样本d的一个输入分量x_i。在式（14-12）的推导过程中用到的是高等数学中复合函数求导的知识。在误差函数 $E(\vec{w})$ 的表达式式（14-7）中，对给定的一组样本目标输出t_d为常数，因此有

$$\frac{\partial t_d}{\partial \mathrm{w}_i} = 0$$

而实际输出o_d是W_i的函数，对于线性单元而言，有

$$o_d = \sum_{i=0}^{n} w_i x_{id}$$

在计算o_d对W_i的偏导时，w的其余分量W_j(j=0,1, 2,..., n, j≠i)均可视为常数，对于给定样本，x_{id}亦为常数，故

$$\frac{\partial o_d}{\partial \mathrm{w}_\mathrm{i}} = x_{id}$$

至此我们得到了一个能够用线性单元的输入x_{id}，输出o_d以及训练样本的目标值t_d表示的$\partial E/\partial W_i$，代入式（14-9）得到了梯度下降的权值更新法则。

$$\Delta w_i = \eta \sum_{d \in D} (t_d - o_d) x_{id} \qquad (14\text{-}13)$$

综上所述，训练线性单元的梯度下降算法如下。随机选取一个初始权向量；计算所有训练样本经过线性单元的输出，然后根据式（14-13）计算每个权值的ΔW_i，通过式（14-10）来更新每个权值，然后重复这个过程。该算法的伪代码描述如算法14.1所示。

算法**14.1** 训练线性单元的梯度下降算法

GradDesc(trainset, η)

{//trainset中每一个训练样本以序偶<$\vec{x}$,t>的形式给出，其中$\vec{x}$是样本特征向

//量，是系统的输入；t是目标输出值，通常是类标签的某种编码，η是学习率。

将每个网络权值W_i初始化为某个小的随机值;

遇到终止条件之前，重复以下操作:

初始化每个ΔW_i为0;

对于训练集合trainset中的每个<$\vec{x}$,t>，做:

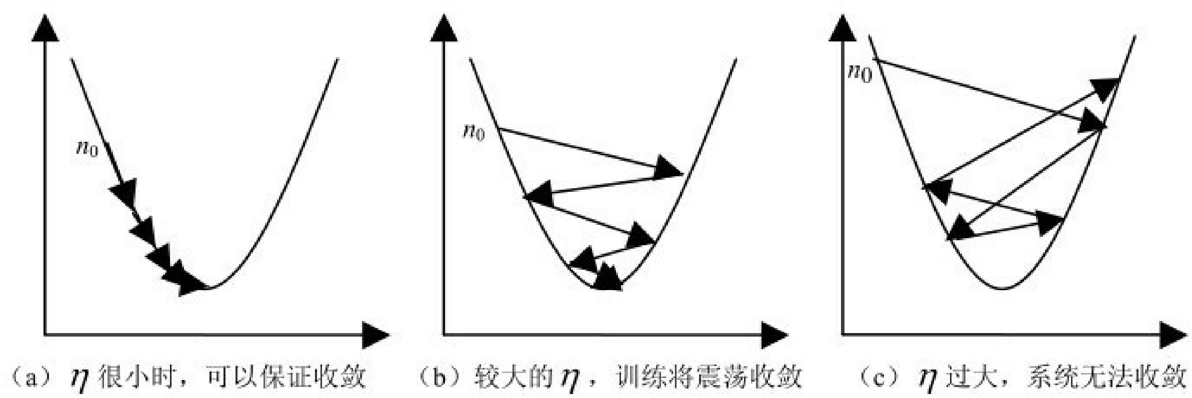
把样本特征向量$\vec{x}$作为线性单元的输入，计算输出o;

对于线性单元的每个权w，$_i$做$\Delta W_i \leftarrow + -\Delta W_i \eta t\ o x(\ )_i$;

对于线性单元的每个权W_i，做$W_i \leftarrow W_i + \Delta W_i$;

}

由于二次误差曲面仅包含一个全局最小值，算法14.1 最终会收敛到具有最小误差的权向量，但必须使用一个足够小的学习率η。如果η太大，梯度下降搜索就有越过误差曲面最小值而不是停留在那一点的危险，如图14.7所示。一种好的改进策略是随着梯度下降步数的增加而逐渐减小η的值。



▲图14.7 η对于梯度下降搜索收敛的影响（n0为搜索起始点）

5．增量梯度下降（**incremental gradient descent**）

很多人工智能和模式识别的问题最终都可转化为一个求最优的问题，而梯度下降法作为一种重要寻优手段，适用于满足以下条件的任何情况。

● 搜索的假设空间包含连续参数化的假设（线性单元梯度下降搜索中的解w是连续变化的）。

●误差（$E(\vec{w})$）对于这些假设参数（w）可微。

应用梯度下降的主要实践问题如下。

● 有时收敛过程可能非常慢（需要上千次的迭代）。

● 如果在误差曲面上存在多个局部极小值，算法不保证能够找到全局最小值（与初始搜索位置有关）。

为缓解这些问题，人们提出了增量梯度下降法，又名随机梯度下降。

标准梯度下降在对训练集D中的所有样本的平方误差求和后计算权值更新，式（14-13）中对所有D中样本的Σ求和说明了这一点；而增量梯度下降法根据每个单独样本的误差增量计算权值更新，得到近似的梯度下降搜索。修改后的训练法则与式（14-13）相似，只是在迭代计算每个训练样本时根据下面的公式来更新权值。

$$\Delta w_i = \eta(\mathrm{t}-\mathrm{o})x_i \qquad (14\text{-}14)$$

增量梯度下降法的伪代码描述如算法14.2所示。

算法**14.2** 训练线性单元的增量梯度下降算法

IncGradDesc(trainset, η)

{//trainset中每一个训练样本以序偶<$\vec{x}$,t>的形式给出，其中$\vec{x}$是样本特征向

//量，是系统的输入；t是目标输出值，通常是类标签的某种编码，η是学习率。

将每个网络权值W_i初始化为某个小的随机值;

遇到终止条件之前，重复以下操作:

初始化每个ΔW_i为0;

对于训练样本trainset中的每个<

,t>，做:$\vec{x}$把样本特征向量$\vec{x}$作为线性单元的输入，计算输出o;

对于线性单元的每个权W_i，做$W_i \leftarrow W_i+\eta(t-o)\ x_i$;

}

增量梯度下降可以被看作是为每个单独的训练样本d定义不同的误差函数$E_d(\vec{w})$ 。

$$E_d(\vec{w})=\frac{1}{2}(t_d-o_d)^2 \qquad (14\text{-}15)$$

在每次迭代中按照关于$E_d(\vec{w})$的梯度来改变权值。在迭代完所有训练样本时1轮时，这些权值更新的序列给出了对原来误差函数$E(\vec{w})$的标准梯度下降的一个合理近似。只要η足够小，增量梯度下降可以以任意程度接近标准梯度下降。

提示

标准梯度下降在权值更新前对所有样本汇总误差，而增量梯度下降的权值是通过考察每个训练样本来更新的，如果$E(\vec{w})$存在多个局部极小值，增量梯度下降有时可避免陷入这些局部极小值，因为它使用$E_d(\vec{w})$而不是$E(\vec{w})$来引导搜索。

[例14.2] 寻优方法比较。

下面给出了除梯度下降外的两种常用优化方法：偏导数和二次规划。思考为什么在ANN的训练过程中能否选择这两种方法。

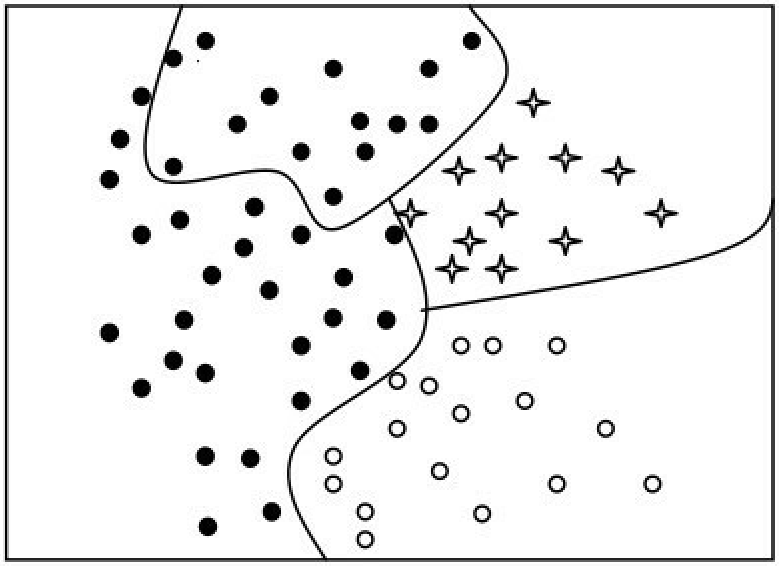
（1）由于我们可以写出$\partial E/\partial W_i$的表达式，一个可能会产生的疑问是是否可以采用偏导数为 0的方法来解决误差函数$E(\vec{w})$的优化问题，毕竟这是高等数学中有关多元函数极值和最值的一种经典方法，可能也是读者们面对优化问题首先能够想到的解决方法。然而，这里我们要面对的通常是含有大量权值的系统，它对应着维数非常高的空间中的误差曲面（每个权值1维），而每1维都有多个可能的局部极小值点（对该维偏导数为 0），这样总的候选局部极小值数目就是各个维上候选局部极小值的个数的乘积，这将是一个庞大的数目。

也就是说如果通过这种方法解决误差函数的优化问题，设共有n个权值，那么首先要解一个由n 个方程$\partial E/\partial W_i=0$, (i=1, 2,…, n)组成的联立方程组，我们姑且不说这个方程组的可解性如何，假设第i维上满足$\partial E/\partial W_i=0$的解的个数为$n_i(i=1,2,\dots,n)$，则总的候选局部极小值个数将是$\prod_{i=1}^{n} n_i$，还要再从这么多候选中找出一个全局的极小值仍将是个十分棘手的任务。

（2）对于线性单元，学习权向量的又一种可行的方法是线性规划（Linear programming）。线性规划是解线性不等式方程组的一种通用的有效方法。注意在线性单元训练中，每一个训练样本对应一个形式为$\vec{w}\cdot\vec{x}>0$或$\vec{w}\cdot\vec{x}\leq 0$的不等式，且它们的解就是我们期望的权向量。遗憾的是，这种方法仅当训练样本线性可分时有解。即便是在对它进行改进之后也不能扩展到训练多层网络。相反，正如14.2.2小节将要讨论的，梯度下降算法可以被简单地扩展到多层网络，用于反向传播算法的学习之中。

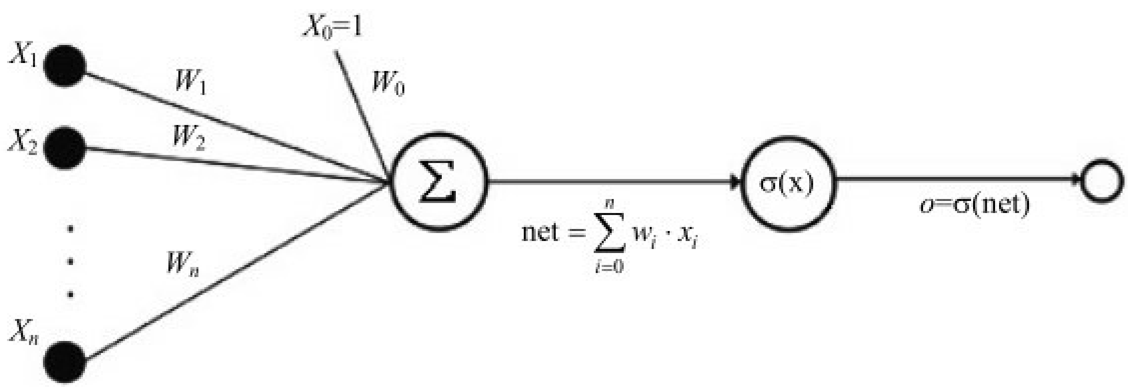
14.2.2 多层人工神经网络

利用前面学过梯度下降法训练线性单元，我们只能够得到一个最佳拟合训练数据的线性超平面（w，b），这里 $b=W_0$。然而线性决策面的分类能力有限，难以胜任很多复杂的分类任务（如汽车自动驾驶、光学字符识别、人脸识别等）。多层人工神经网络能够表示种类繁多的非线性曲面，得到高度非线性化的决策区域，如图14.8所示。



▲图14.8 一个4类问题的高度非线性化的决策区域

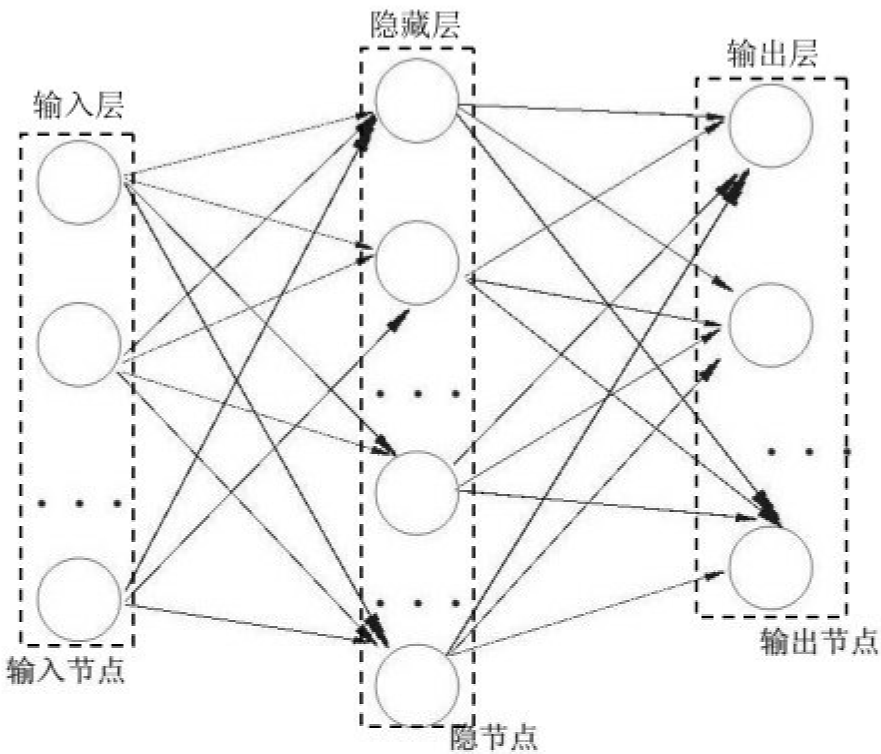
典型的ANN神经元如图14.9所示。输入信号$\vec{x}$经过一个累加器累加（整合）后的信号$\vec{w}\cdot\vec{x}$被送入一个激活函数σ，从而得到该神经元的输出o。而这个神经元的输出o又可以作为下一个或多个神经元的输入。



▲图14.9 人工神经网络单元

大脑中神经细胞和其他神经细胞是相互连接在一起的，人工神经元也可以按照类似的方式连接从而组成人工神经网络。一种最广泛使用的连接方式就是如图 14.10 所示的分层前馈网络（feedforward network），

每一层神经元的输出都前馈至它们的下一层（图14.10中为右侧的一层），直至获得整个网络的输出。



▲图14.10 三层BP神经网络的结构示意

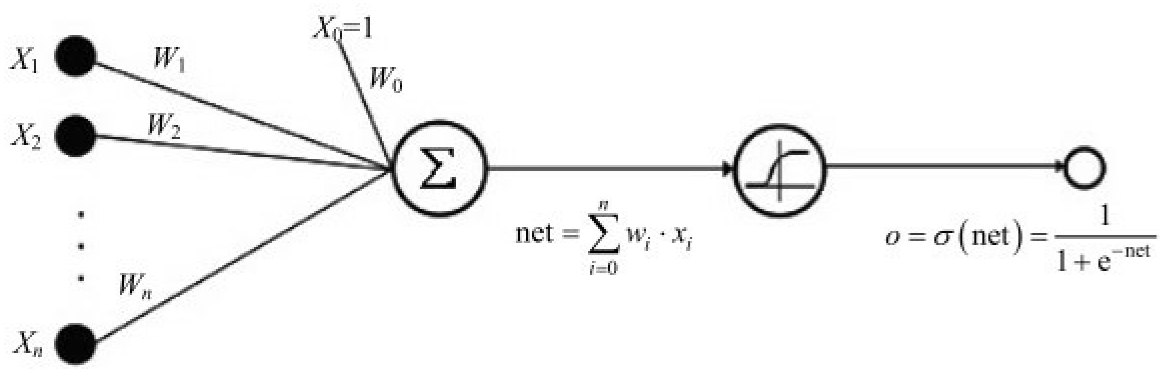
在分层网络中，一般至少有3个层：一个输入层、一个输出层还有一个或多个隐藏层。相邻层之间的单元是全连接的，即输入层中的每个输入都连接到了隐藏层的每一个神经元，而隐藏层的每个神经元的输出都连接到输出层的每一个神经元。多层网络可以解决单层网络所无法解决的问题，如非线性分类、精度极高的函数逼近等。只要有足够多的神经元，这些都可以实现。

14.2.3 sigmoid单元

我们希望得到形式更为复杂的非线性决策区域，但多个线性单元的

连接仍产生线性函数，因此将多个线性单元连接成网络的方法显然是不可取的。这样就需要输出是输入的非线性函数的单元。同时从仿生学的角度来考虑，ANN的神经元也应当像生物神经元那样存在某种刺激。

于是我们采用某种非线性激励函数作用于单元的净输入，然后以非线性激励的响应（激励函数的输出）作为神经元的输出。此外，为了能够在网络训练过程中应用梯度下降算法，要求输出必须是输入的可微函数。满足这些条件的一个理想选择是如图14.11所示的sigmoid单元，它以著名的sigmoid函数作为激励函数。



▲图14.11 神经网络中的 sigmoid单元

我们仍旧像在线性单元中所做的那样，首先计算它的n个输入的线性组合并且加上一个偏置W_0，而后将sigmoid函数作用于这个结果，最终以sigmoid函数的输出作为单元输出o，从而实现了输入到输出的非线性映射。形式化地讲

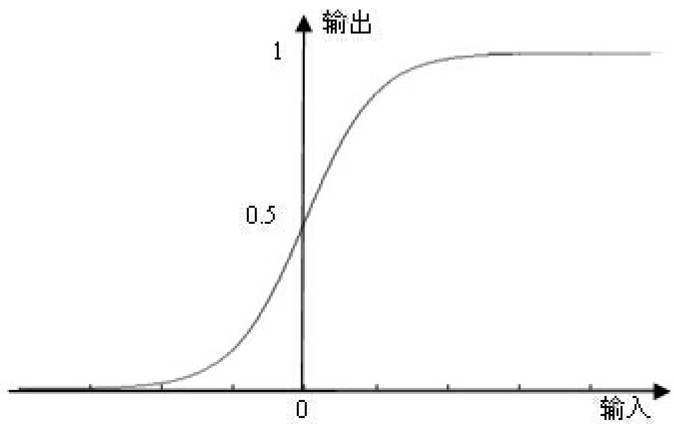
$$o = \sigma(\vec{w} \cdot \vec{x}) \qquad （14\text{-}16）$$

式中：$\vec{w}$ =(W_0,W_1,W_2,...,W_n)；$\vec{x}$ =(1,x_1,x_2,...,x_n)。

而sigmoid函数

$$\sigma(y)=\frac{1}{1+e^{-y}} \tag{14-17}$$

sigmoid函数的定义域为[−∞,+∞]，值域为[0，1]，是一个单调递增的平滑函数。函数波形如图14.12所示。



▲图14.12 sigmoid函数曲线

由于sigmoid函数将非常大的输入值范围映射到一个小范围的输出，它常被称为sigmoid单元的挤压函数。该函数还有一个很好的特性，即它的导数很容易用它的输出来表示。

$$\frac{\mathrm{d}\sigma(y)}{\mathrm{d}y}=\sigma(y)(1-\sigma(y)) \tag{14-18}$$

推导过程如下。

$$\frac{d\sigma(y)}{dy}=\frac{d}{dy}(\frac{1}{1+e^{-y}})=-\frac{1}{(1+e^{-y})^2}\cdot\frac{d}{dy}(e^{-y})$$
$$=\frac{e^{-y}}{(1+e^{-y})^2}=\frac{1}{1+e^{-y}}\cdot\frac{e^{-y}}{1+e^{-y}}$$
$$=\frac{1}{1+e^{-y}}\cdot(1-\frac{1}{1+e^{-y}})=\sigma(y)(1-\sigma(y))$$

式（14-18）无疑将大大简化梯度下降算法中的导数计算。

14.2.4 反向传播（BP，back propogation）算法

将多个sigmoid单元互相连接形成如图14.10所示的3层网络，反向传播算法可用来学习这个网络的权值。它仍采用梯度下降算法，以最小化网络实际输出与目标输出之间的平方误差为目标。不同的是，这里的输出是整个网络的输出，而不再是单个单元的输出，所以有必要重新定义误差E，以便对网络输出层的所有单元误差求和。

$$E(\vec{w})=\frac{1}{2}\sum_{d\in D}\sum_{k\in outputs}(t_{kd}-o_{kd})^2 \qquad （14-19）$$

其中outputs是网络输出层单元的集合，t_{kd}和o_{kd}是将训练样本d作为网络输入时在第k个输出层单元的输出值，误差E被看成是网络各层之间所有连接的权的函数，由这些权值共同决定。

反向传播算法需要在一个巨大的解空间中搜索能够使式（14-19）最小化的$\vec{w}$，这里$\vec{w}$表示网络各层之间所有连接的权值组成的集合，这个空间由网络中所有单元的所有可能权值来定义。此时的误差曲面不再是图14.6中的二次误差曲面（超抛物面），它更加复杂并且可能有多个局部极小值。sigmoid单元的可微性使得我们仍然可以借助梯度下降法来优化$E(\vec{w})$，但此时梯度下降仅能保证收敛到局部最小值，而未必是

全局最小值。

14.2.1小节中给出了训练线性单元的标准梯度下降和增量梯度下降算法的推导和伪代码描述，下面将针对如图14.10所示的包含2层sigmoid单元且层与层单元之间全连接的网络给出训练的增量梯度下降算法推导以及其伪代码描述。

1．反向传播算法的推导

首先引入以下记号。

x_{ji}——单元j的第i个输入

W_{ji}——与输入x_{ji}相关联的权值

$net_j=\Sigma_i W_{ji}x_{ji}$——单元j的净输出（输入的加权和，未经过激励函数）

$o_j=\sigma(net_j)$——单元j的实际输出

t_j——单元j的目标输出

σ——sigmoid函数

outputs——输出层单元的集合

增量梯度下降法的主要特点是对于每个训练样本d，利用关于这个样本的误差e_d的梯度来修改权值。换言之，对于每个训练样本d每个权值W_{ji}被增加ΔW_{ji}。

$$\Delta w_{ji} = -\eta \frac{\partial E_d}{\partial w_{ji}} \tag{14-20}$$

其中，e_d是训练样本d的误差，通过对输出层所有单元的求和得到。

$$E_d(\vec{w}) = \frac{1}{2}\sum_{k \in outputs}(t_k - o_k)^2 \tag{14-21}$$

这里 t_k是输出层单元 k对于训练样本d的目标输出值，o_k是以训练样本d作为输入时第 k个输出层单元的实际输出。

现在摆在我们面前的问题是要导出一个$\partial e_d/\partial W_{ji}$的表达式。注意到权值$W_{ji}$仅能通过$net_j$影响网络的其他部分，因此可利用复合函数的求导法则得到。

$$\frac{\partial E_d}{\partial w_{ji}} = \frac{\partial E_d}{\partial net_j}\frac{\partial net_j}{\partial w_{ji}} = \frac{\partial E_d}{\partial net_j}\frac{\partial(\sum_i w_{ji}x_{ji})}{\partial w_{ji}} = \frac{\partial E_d}{\partial net_j}x_{ji} \tag{14-22}$$

下面分两种情况导出 $\frac{\partial E_d}{\partial net_j}$ 的表达式。

（1）单元j是一个输出层单元

由于net_j仅能通过o_j影响网络，所以可再次利用复合函数的求导法则得。

$$\frac{\partial E_d}{\partial net_j} = \frac{\partial E_d}{\partial o_j}\frac{\partial o_j}{\partial net_j} \tag{14-23}$$

考虑式（14-23）的第1项。

$$\frac{\partial E_d}{\partial o_j} = \frac{\partial}{\partial o_j}(\frac{1}{2}\sum_{k\in outputs}(t_k - o_k)^2)$$

注意到当k≠j时：$\frac{\partial}{\partial o_j}(t_k - o_k)^2 = 0$，因此

$$\begin{aligned}\frac{\partial E_d}{\partial o_j} &= \frac{\partial}{\partial o_j}\frac{1}{2}(t_j - o_j) \\ &= \frac{1}{2}2(t_j - o_j)\frac{\partial (t_j - o_j)^2}{\partial o_j} \\ &-(t_j - o_j)\end{aligned} \tag{14-24}$$

接下来考虑式（14-23）的第2项。

由于$o_j=\sigma(net_j)$，所以$\partial o_j/\partial net_j$就是sigmoid函数的导数，根据式（14-18），sigmoid函数的导数为$\sigma(net_j)(1–\sigma(net_j))$。因此有

$$\frac{\partial o_j}{\partial \mathrm{net}_j} = \frac{\partial \sigma(\mathrm{net}_j)}{\partial \mathrm{net}_j} = o_j(1 - o_j) \tag{14-25}$$

将式（14-24）和（14-25）代入式（14-23），得

$$\frac{\partial E_d}{\partial \mathrm{net}_j} = -(t_j - o_j)o_j(1 - o_j) \tag{14-26}$$

将式（14-26）代入式（14-22），再将式（14-22）代入式（14-20）便得到输出单元的权值更新法则。

$$\Delta w_{ji} = -\eta \frac{\partial E_d}{\partial w_{ji}} = \eta(t_j - o_j)o_j(1 - o_j)x_{ji} \tag{14-27}$$

令$\delta_j=(t_j–o_j)\ o_j(1–o_j)$，则式（14-27）可表示为

$$\Delta w_{ji} = \eta \delta_j x_{ji} \tag{14-28}$$

提示

式（14-28）中的δ_j与$-\dfrac{\partial E_d}{\partial net_k}$相等。下文中将使用$\delta_i$来表示任意单

元i的$-\frac{\partial E_d}{\partial net_i}$。

（2）单元j是一个隐藏层单元

对于隐藏层单元的情况，W_{ji}是间接地影响网络输出，从而影响训练误差e_d，因此有必要定义由单元j的输出所能连接到的所有单元的集合Downstream(j)。因为net_j只能通过Downstream(j)中的单元影响网络输出进而影响到e_d，故有如下推导。

$$\begin{aligned}\frac{\partial E_d}{\partial \text{net}_j} &= \sum_{k\in \text{Downstream}(j)} \frac{\partial E_d}{\partial \text{net}_k}\frac{\partial \text{net}_k}{\partial \text{net}_j} \\ &= \sum_{k\in \text{Downstream}(j)} -\delta_k \frac{\partial \text{net}_k}{\partial \text{net}_j}\end{aligned}$$

$$\begin{aligned}&= \sum_{k\in \text{Downstream}(j)} -\delta_k \frac{\partial \text{net}_k}{\partial o_j}\frac{\partial o_j}{\partial \text{net}_j} \\ &= \sum_{k\in \text{Downstream}(j)} -\delta_k w_{kj} \frac{\partial o_j}{\partial \text{net}_j} \\ &= \sum_{k\in \text{Downstream}(j)} -\delta_k w_{kj} o_j (1-o_j) \\ &= -o_j(1-o_j)\sum_{k\in \text{Downstream}(j)} \delta_k w_{kj}\end{aligned}$$

同样令$\delta_j = -\frac{\partial E_d}{\partial \text{net}_j}$，即可得到隐藏层单元j的权值更新法则。

$$\Delta w_{kj} = \eta \delta_j x_{ji} \qquad （14\text{-}29）$$

2．反向传播算法的训练过程

首先创建1个含有1个隐藏层的网络，并随机地为这些神经细胞的权

重赋一个很小的实数值，如–0.05～0.05之间的数（具体原因参见14.2.5小节）。然后把1个输入样本向量送入网络的输入端（一般情况下网络输入单元的数目等于输入样本特征向量的维数），并计算网络输出值。求得这一输出值和训练样本目标输出值之间的平方误差 e_d。利用这一误差值就可以来调整来输出层单元的权值，使得当同样的输入再次送入网络时，其输出能向正确答案接近一些。一旦输出层的权值已经调整完毕，就可以对隐藏层做同样的事情。上述过程要对训练集中的所有不同的输入样本向量重复进行许多次（经常将一次重复称为一个时代——Epoch），直到误差值降低到所处问题可以接受的一个阈值之内。这时我们说网络已经训练好了。

反向传播算法的伪代码描述如算法14.3所示。

算法**14.3** 包含两层**sigmoid**单元的人工神经网络的反向传播算法（增量梯度下降版本）

BackPropagation（trainset, η, n_{in}, n_{out}, n_{hidden}）

{//trainset中每一个训练样本以序偶<$\vec{x}$,t>的形式给出，其中$\vec{x}$是样本特征向量，是系统的输入；

// η是学习速率（例如0.1或0.05）n_{in}是网络输入单元的数量；n_{out}是输出层单元数量；

//$n_{hidden/}$是隐藏层单元数量。x_{ji}表示从单元i到单元j的输入，W_{ji}表示从单元i到单元j的权值

创建具有个n_{in}输入，n_{hidden}个隐藏层单元，n_{out}个输出层单元的网络;

将所有的网络权值初始化为小的随机值（如−0.05～0.05的数）;

在遇到终止条件之前，重复以下操作:

对于训练样本集合trainset中的每个<$\vec{x},\vec{t}$>:

//将输入沿网络向前传播

把样本$\vec{x}$输入网络，并计算网络中每个单元u的输出o_u;

//使误差沿网络反向传播

对于网络中的每个输出单元k，计算它的误差项δ_k;

$$\delta_k \leftarrow o_k(1-o_k)(t_k-o_k) ;$$

对于网络中的每个隐藏单元h，计算它的误差项δ_h;

$$\delta_h \leftarrow o_h(1-o_h)\sum_{k \in outputs} w_{kh}\delta_k ;$$

更新每个网络权值W_{ji};

$$w_{ji} \leftarrow w_{ji}+\Delta w_{ji}，\text{其中 } \Delta w_{ji}=\eta\delta_j x_{ji} ;$$

}

14.2.5 训练中的问题

1．收敛性和局部极小值

反向传播算法在解空间中寻找能够最小化训练误差的网络权值。对于含有非线性Sigmoid单元的多层网络，误差曲面可能含有多个不同的局部极小值，梯度下降搜索有可能陷入到这些局部极小值中。因此，反向传播算法仅能保证收敛到误差 E 的某个局部极小值，而不一定收敛到全局最小误差。

通过以下方法可以有效降低搜索停留在局部极小值的概率，使反向传播算法尽可能地收敛到全局最小误差。

（1）将网络权值初始化为接近于0的小随机值

注意到sigmoid函数在其输入接近0时接近线性（图14.12）。如果把网络权值初始化为接近于0的值（−0.05～0.05之间的数），则作为

sigmoid单元净输入的net也必然接近于0，因此在早期的梯度下降步骤中，网络表现为一个非常平滑的函数，近似为输入的线性函数，基本不存在局部极值的问题。当训练进行一定时间后，随着权值的增长，网络演变为可以表示高度非线性的函数，从而开始出现更多的局部极小值，但一般情况下此时搜索已经足够接近全局最小值，即便是这个区域的局部最小值也是可以接受的。

（2）增加冲量项

修改算法 14.3 中的权值更新法则，使本次的权值的更新部分地依赖于上一次迭代时的更新，形式如下。

$$\Delta w_{ji}(n) = \eta \delta_j x_{ji} + \alpha \Delta w_{ji}(n-1) \qquad (14\text{-}30)$$

其中，$\Delta W_{ji}(n)$表示算法14.3主循环中的第n次迭代时的权值更新，而$0 \le \alpha < 1$是一个称为冲量（momentum）的常数。式（14-30）右侧的第1项是算法14.3中的权值更新法则，而第2项是新增的冲量项。

冲量项有时可以带动梯度下降搜索冲过狭窄的局部极小值而不是陷入其中。设想一个球沿误差曲面向下滚，α的作用是增加冲量，使这个球从一次迭代到下一次迭代时以同样的方向滚动。冲量有时会使这个球滚过误差曲面的局部极小值或平坦区域。同时，冲量项还具有在梯度不变的区域逐渐增大搜索步长，从而可以加快收敛的作用。

（3）使用随机的梯度下降代替真正的梯度下降

算法 14.3 采用的梯度下降的随机近似对于每个训练样例沿一个不同的误差曲面有效下降，它依靠这些梯度的平均来近似对于整个训练集合的梯度。这些不同的误差曲面通常有不同的局部极小值，这使得下降过程不太可能陷入某一个局部极小值。

（4）在搜索过程中使用逐渐减小的学习率

较大的学习率η能够在训练初期加快收敛进程，然而对于复杂的误

差曲面，较大的η常常使搜索有越过误差曲面最小值而不是停留在那一点的危险。因此，可以在随着训练的进行，逐步减η，保证误差的充分减小（参见14.5.1小节）。

（5）尝试不同初始位置的多次搜索

使用同样的训练集训练多个网络，但用不同的随机值初始化每个网络权值。如果不同的训练产生不同的局部极小值，那么选择对于独立的测试集合分类性能最好的网络。

2．训练的终止判据

算法14.3中并没有明确指出算法迭代的终止条件，通常有以下3种常用标准可以作为网络训练结束的终止判据。

（1）在迭代的次数到了一个固定值时停止。

（2）在训练样例上的误差降到某个阀值以下时停止。

（3）在独立的测试样本集合上的分类误差符合某个标准时。

终止判据的选择很是重要，因为在典型的应用中，反向传播算法的权值更新迭代会被重复上千次，一味地增加迭代次数很可能无法有效地降低误差，过多的迭代次数还会导致对训练数据的过度拟合。

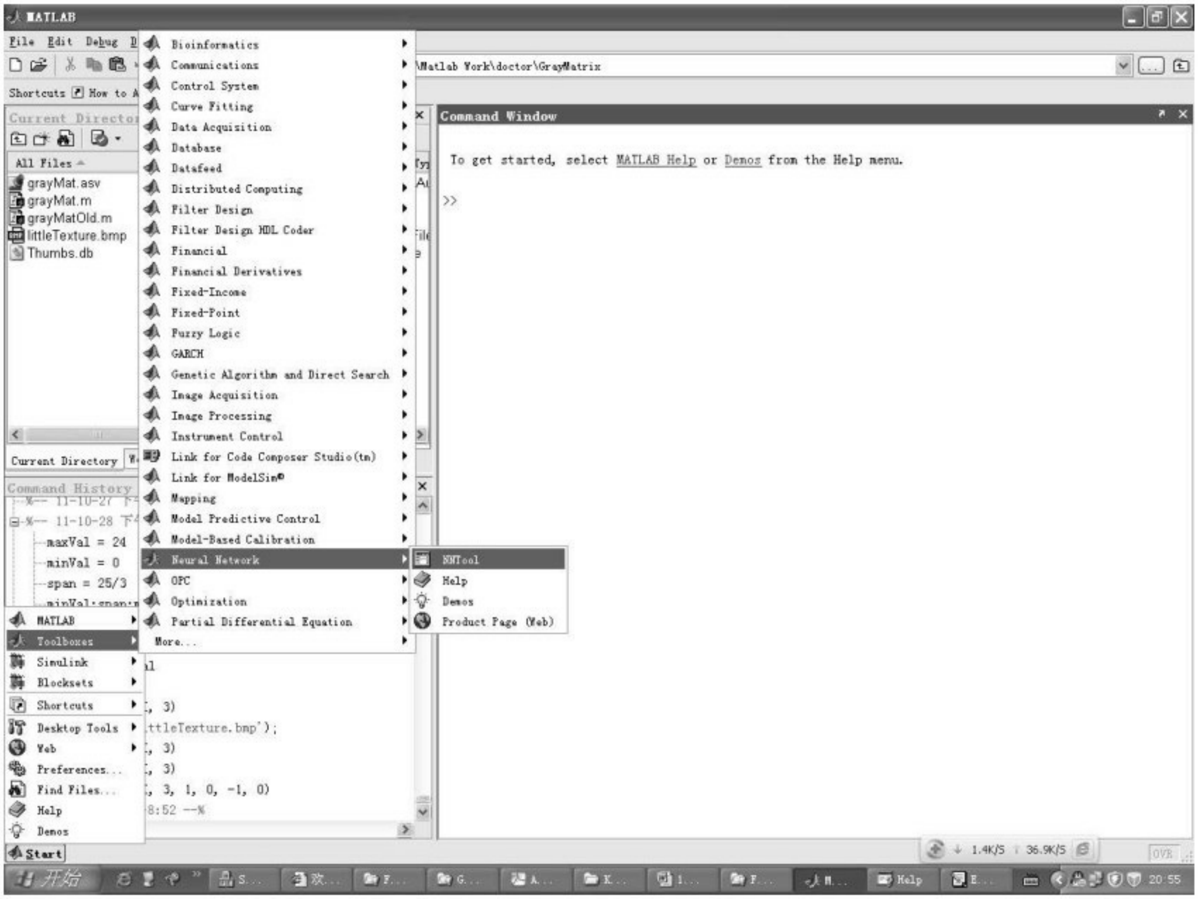
14.3 神经网络算法的可视化实现

Matlab神经网络工具箱（Neural Network Toolbox）中的NNTool函数为我们提供了一种集神经网络系统设计、仿真和分析功能于一体的可视化工具。读者完全不需要编程，且设计和仿真的结果完全能够满足一般科研人员与工程设计人员的需求。

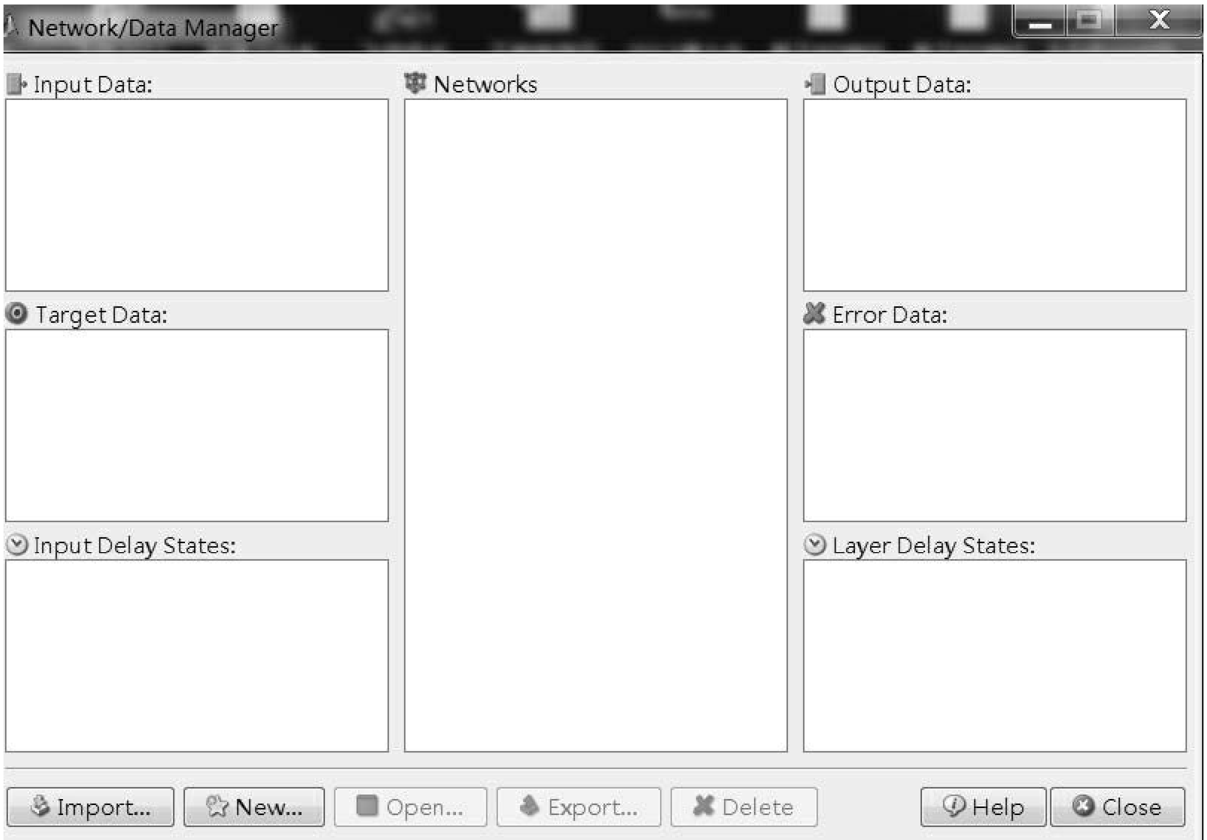
14.3.1 NNTool的主要功能及应用

打开Matlab R2011a应用软件后，单击左下角的“Start”按钮，在弹出

的菜单中选择“Toolboxes”→“Neural Network”中的NNTool图标（见图14.13），或直接在命令窗口键入nntool，即可进入如图14.14所示的神经网络设计主界面。



▲图14.13 以菜单方式打开NNTool

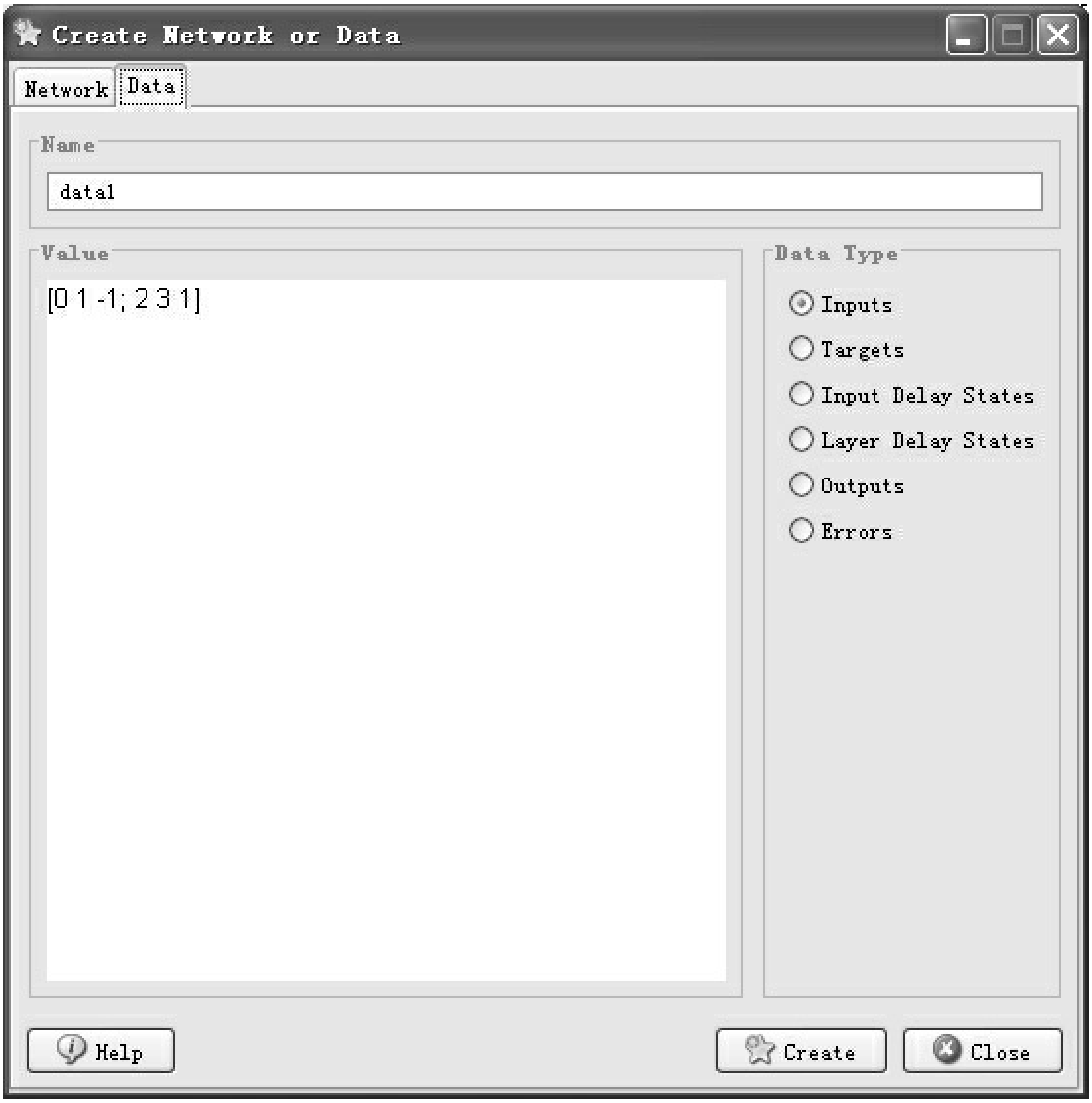


▲图14.14 NNTool主界面

1．训练样本数据的导入

在NNTool中有两种途径可生成训练样本数据。

（1）单击“New”按钮，在弹出窗口中选择“Data”选项卡，直接输入数据，如图14.15所示。



▲图14.15 直接输入数据

界面中各个单选按钮的含义如下。

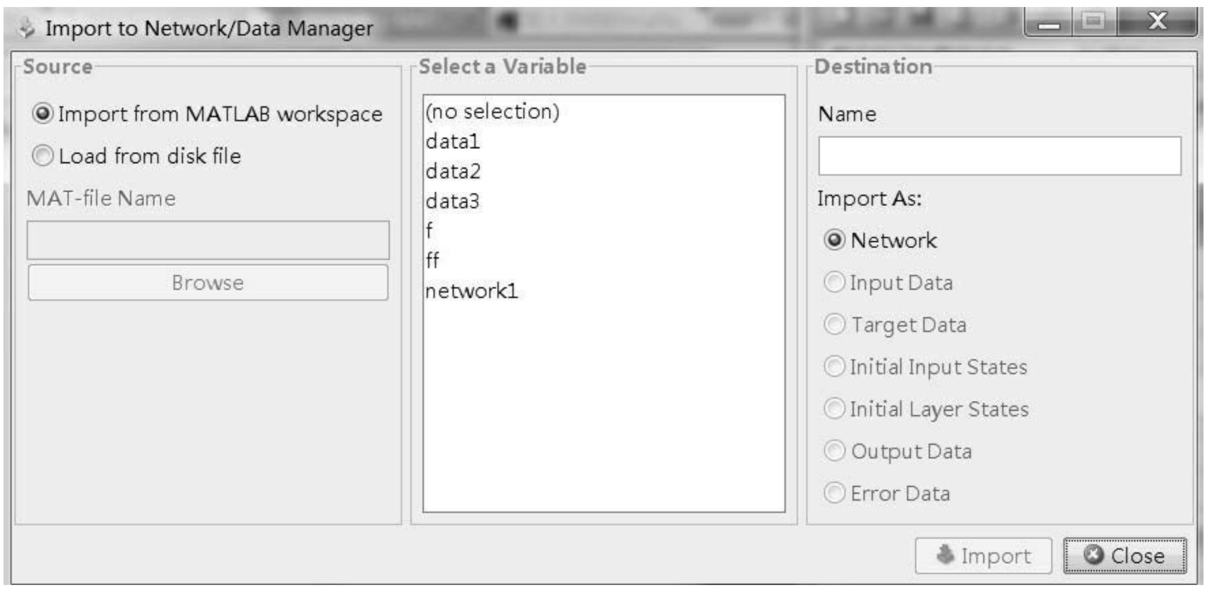
- Inputs：输入值。
- Targets：目标输出值。
- Input Delay States：输入值欲延迟时间。
- Networks：已建构的网络。
- Outputs：输出值。
- Errors：误差值。

● Layer Delay States：输出值欲延迟时间。

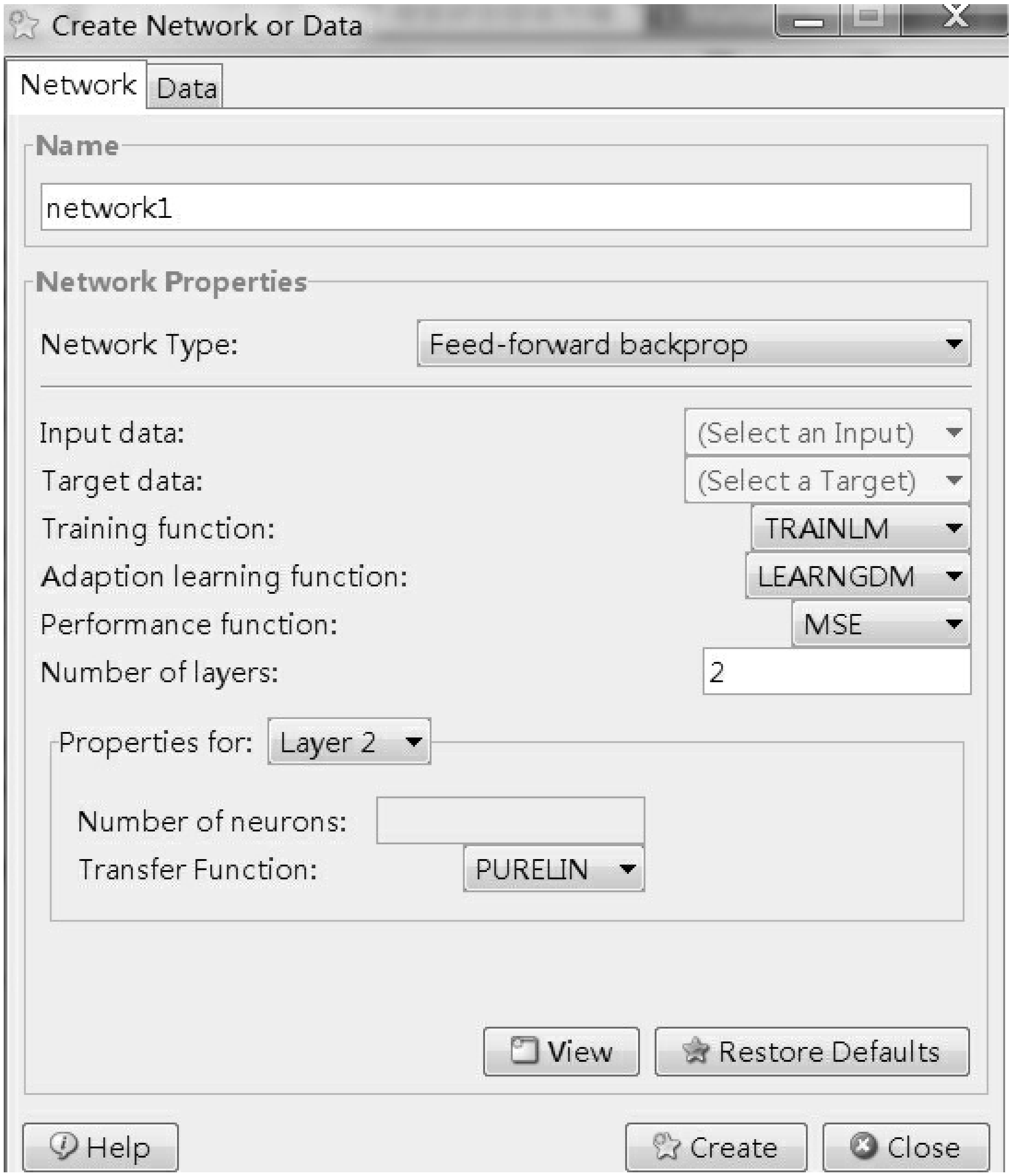
（2）单击“import”按钮，从工作空间或.mat文件中导入相应数据，如图14.16所示。

2．神经网络的创建

单击图 14.13 中的“New”按钮，进入“Network”选项卡，在此选项卡中可对网络的名称、类型、结构和训练函数等进行设置，如图14.17所示。



▲图14.16从工作空间或.mat文件中输入数据



▲图14.17创建神经网络界面

图14.17中的各个可选项的含义如下。

● Name：输入网络名称，如networkl。

● Network Type：网络类型，如Feed-forward backprop（前馈反向传播网络）。

● Input data：选择输入数据，Data选项卡中已经输入。

● Target data：选择目标输出，Data选项卡中已经输入。

● Training function：训练函数，主要指训练迭代中权值的更新方式，其中TRAINGDM是带动量的梯度下降法，TRAINLM是指L-M优化算法，TRAINSCG是指量化共轭梯度法，除此之外还有TRAINGDX、TRAINGDA等。

● Adaption learning function：适应性学习函数，如LEARNGDM（具动量的梯度下降法）。

● Performance function：性能函数，如MSE（均方误差）。

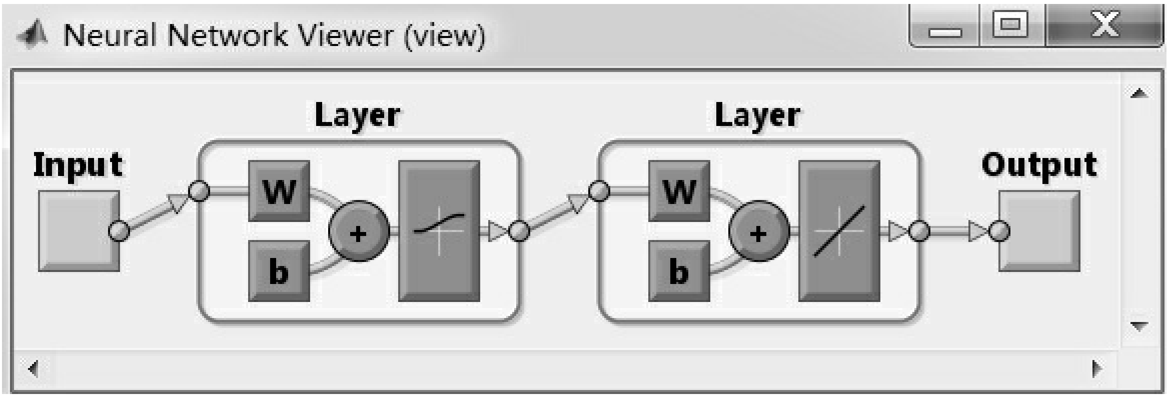
● Number of layers：网络出输入层以外的层数，如 2（隐藏层和输出层）。

● Properties for：由下拉式选单选取欲进行设定的是网络的哪一层，如Layer 2。

● Number of neurons：在“Properties for”下拉选单中所选择的层中神经元的数目。

● Transfer function：在“Properties for”下拉选单中所选择的层中结点的传递函数，即 14.2节中提到的激励函数。其中的TANSIG和LOGSIG统称Sigmoid函数，LOGSIG是单极性S函数， TANSIG是双极性S函数，也叫双曲正切函数，PURELIN是线性函数。

设置完毕后单击Create即可创建网络。单击View按钮，可查看神经网络的结构示意图，如图14.18所示。



▲图14.18 神经网络结构示意图

3．神经网络的初始化

神经网络创建完之后，网络的权值已进行了初始化，若要对权值和阈值重新编辑，则双击“Network/Data Manager”界面中创建的网络名字“network1”，会出现如图14.19所示的对话框，单击“Initialize Weights”按钮即可。BP网络训练时为了避免陷入局部最优，有时需要对权值重新设置，这在NNTool中变得简单易行。



▲图14.19 更改权重界面

4．神经网络的训练

神经网络创建完成后，单击图14.19上方的Train选项卡，可进入如图14.20所示的训练界面，其中各个变量的意义如下。

show：每格多少训练循环次数会显示训练过程，例如要每隔25次迭代显示结果窗口。

showWindow：训练过程中是否显示UI窗口反馈训练过程中的信息。

showCommandLine：训练过程中是否以命令行的方式反馈训练过程中的信息。

epochs：训练的最大循环次数。一般先设置一个较小的数（如200），然后观察收敛结果。如果结果窗口的收敛曲线衰减较快，则表示之前的参数比较有效，因此可填入2 000或更大的数目使得网络收敛，否则修改之前的参数。

ime：最大的训练所须时间，单位为s。由于一般采用最大的训练迭代次数epochs控制迭代结束，因次可设置此项为Inf，即无时间限制。

goal：性能目标，即网络训练要达到的误差目标。该值与“Performance Function”有关，如果使用默认的MSE，则一般满足“goal×样本数量<0.5”就可以了。

max_fail：最大验证数据失败的次数。

mem_reduc：降低内存需求的系数。

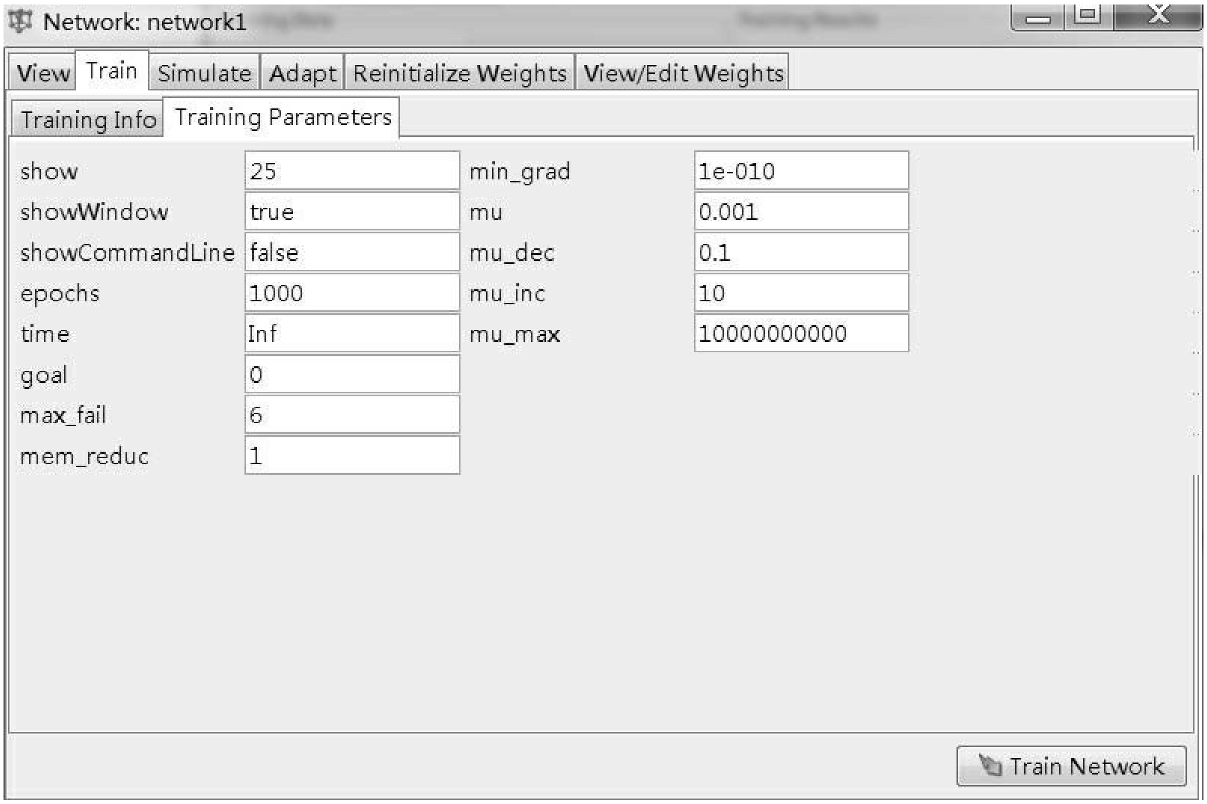
min_grad：最小性能梯度。

mu：动量的初始值。

mu_dec：动量减少系数。

mu_inc：动量增加系数。

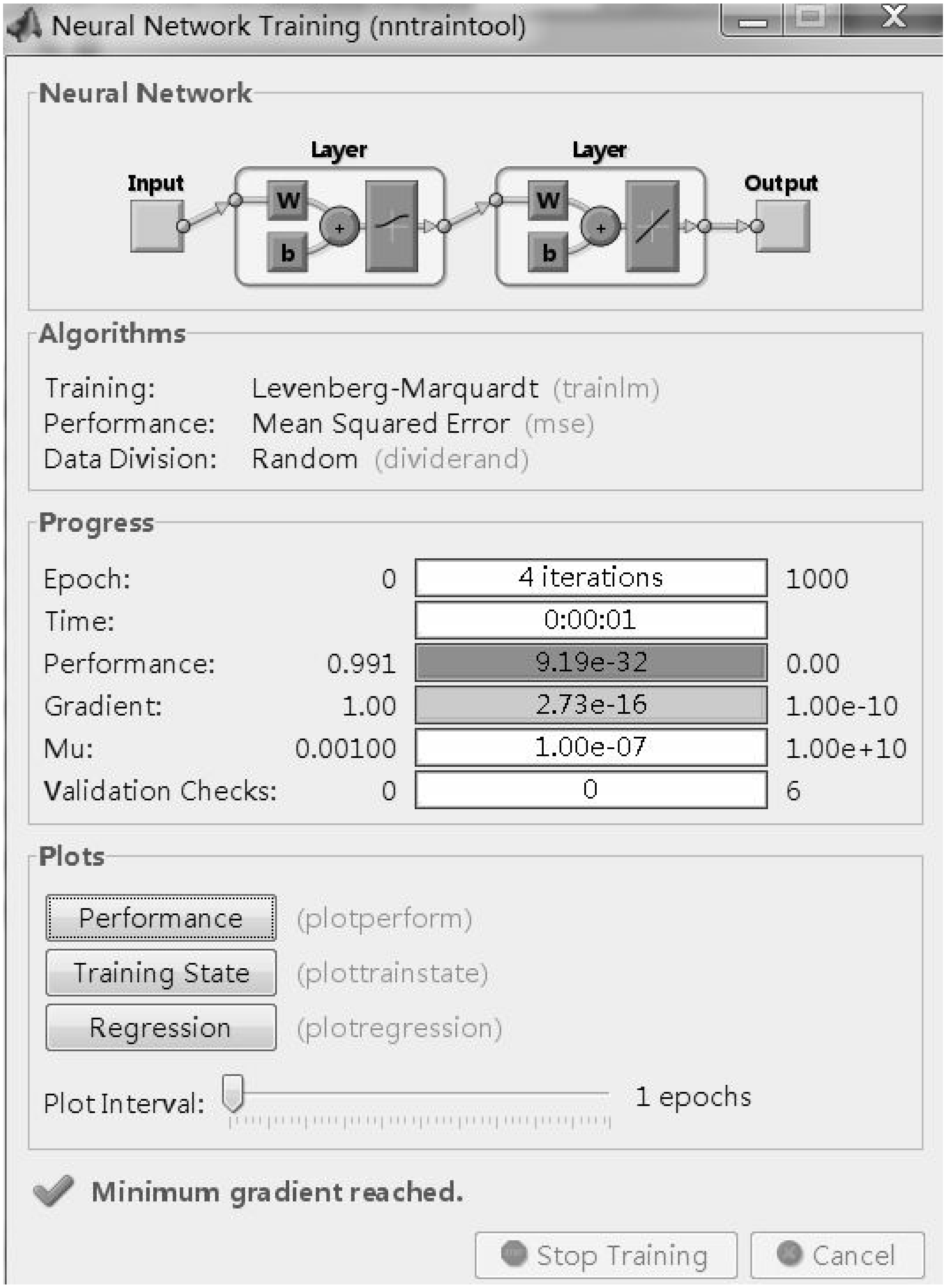
mu_max：动量最大值。



▲图14.20 修改训练参数界面

设置好训练信息Training Info和训练参数Training Parameter后，单击“Train Network”按钮即开始了对网络的训练，过程如图14.21所示。训练结束后，分别单击“Performance”、“Training State”以及“Regression”按钮，可以观察网络训练误差变化曲线、训练状态曲线以及回归曲线。

14.3.2 神经网络的仿真测试



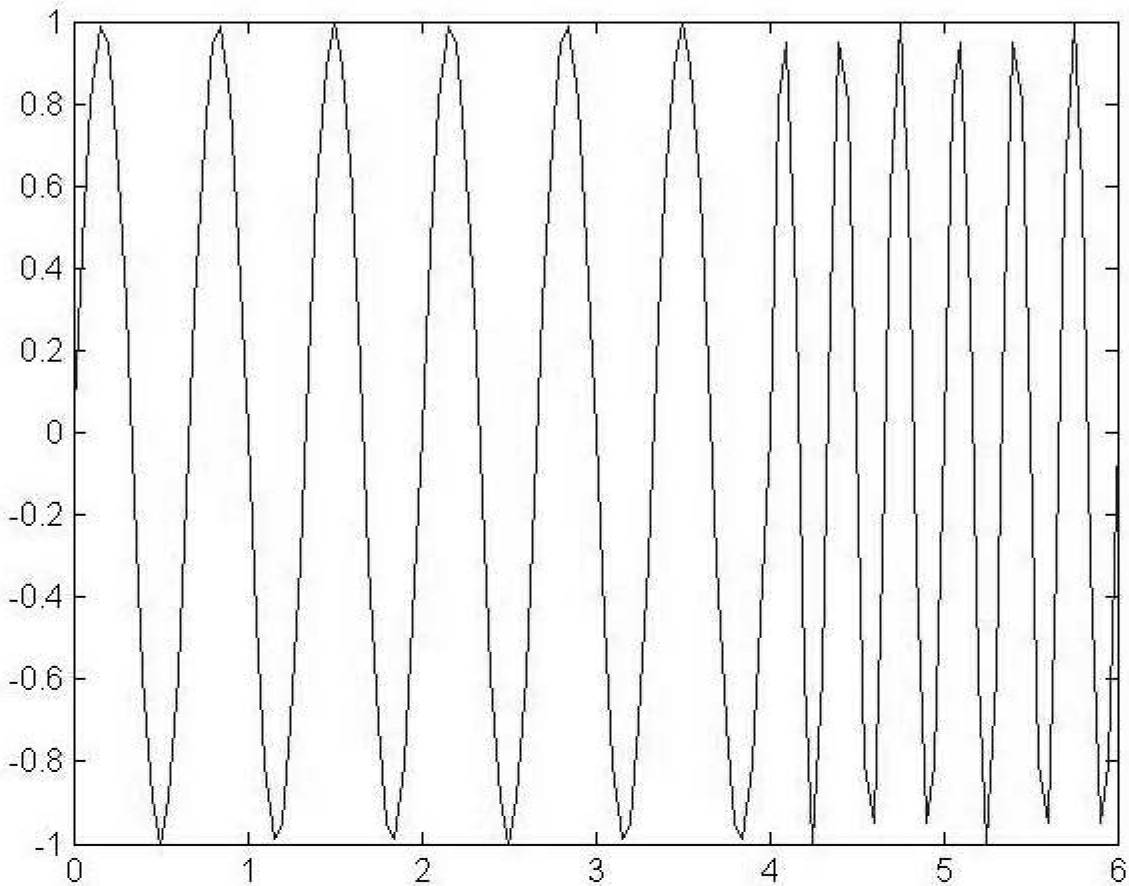
▲图14.21 训练结果图

14.3.2 神经网络的仿真测试

神经网络的仿真测试非常简单，选定训练好的神经网络，单击“View”按钮，再单击图14.20中所示界面上方的“Simulate”选项卡，在“Simulate Data”中的“Inputs”一栏中导入需要测试的数据（需要是工作区的矩阵，所以可以事先将数据写在但单独的程序中，需要时运行程序即可），然后单击“Simulate Network”，测试结束后可以在“Outputs”中看到结果，在“Errors”中可以看到误差。

【例14.3】有一个有频率突变的正弦时变信号，用它来作目标训练一个新建的BP网络，用这个网络去对同频率正弦信号进行识别预测。目标样本信号由下列程序生成，信号波形如图14.22所示。

```
clear
time1=0:0.05:4;
time2=4.05:0.05:6;
time=[time1 time2];
t=[sin(time1*3*pi) sin(time2*6*pi)]; %目标样本
plot(time,t)
```



▲图14.22 目标样本信号

具体步骤如下。

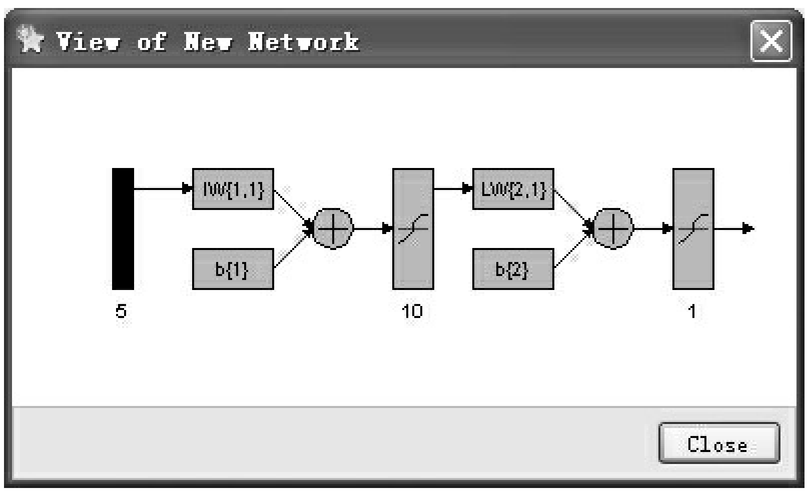
1．在Matlab的命令窗口定义要用到的样本和输入变量。

```
clear
time1=0:0.05:4;
time2=4.05:0.05:6;
time=[time1 time2];
t=[sin(time1*3*pi) sin(time2*6*pi)];    %目标样本
p=delaysig(t,1,5);    %输入样本
p1=randn(size(t))*0.3+t;    %t叠加上一个正态分布的随机噪声
p1=delaysig(p1,1,5);    %仿真输入
```

2．在Matlab的命令窗口用 nntool打开 GUI工具窗口。导入输入样本集 p、测试样本集 p1，导入训练目标t。

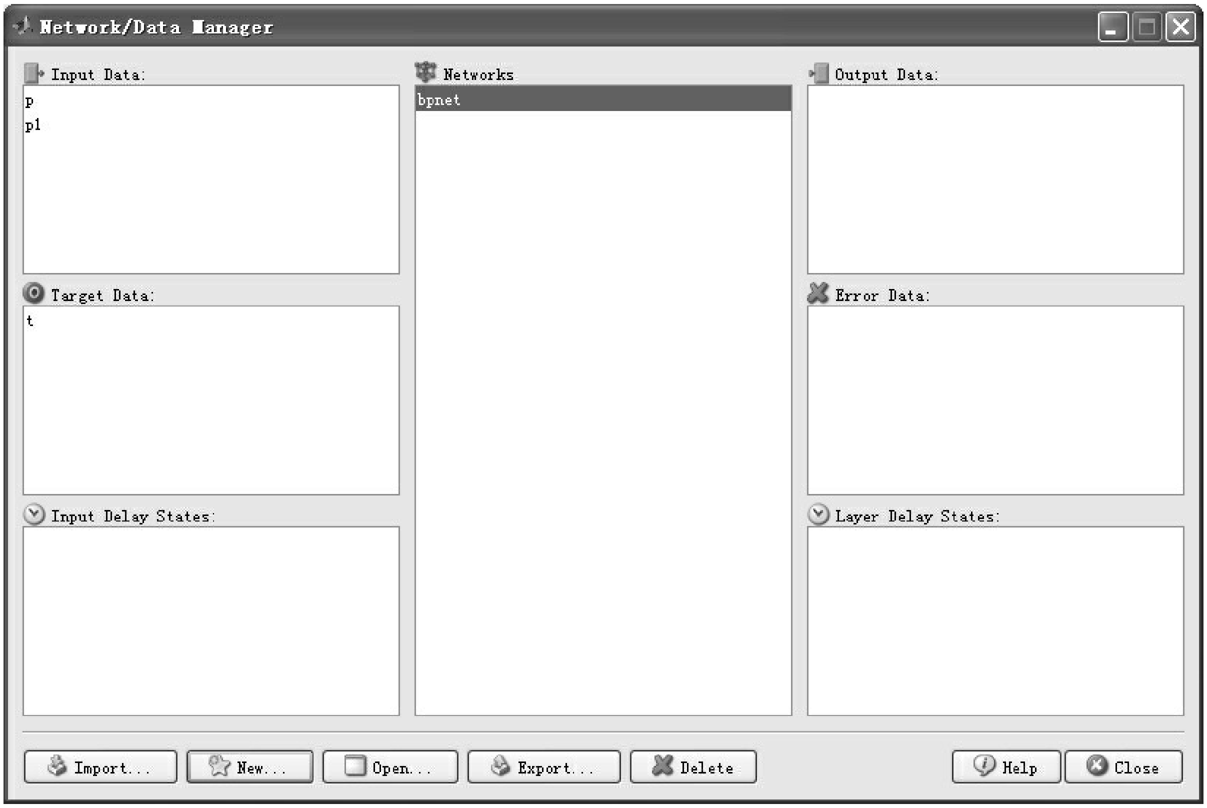
3．在如图 14.14所示的界面中，单击“New”按钮创建新网络。在弹出的“Create Network or Data”对话框中设置网络名为 bpnet。神经网络类型为 Feed-forward backprop，神经网络第一层默认设置（TRAINLM, LEARNGDM, TANSIG）；第二层神经元个数为 1，传递函数为 PURELIN。

4．可以通过单击“View”按钮查看网络结构，如图14.23所示。



▲图14.23 网络结构示意图

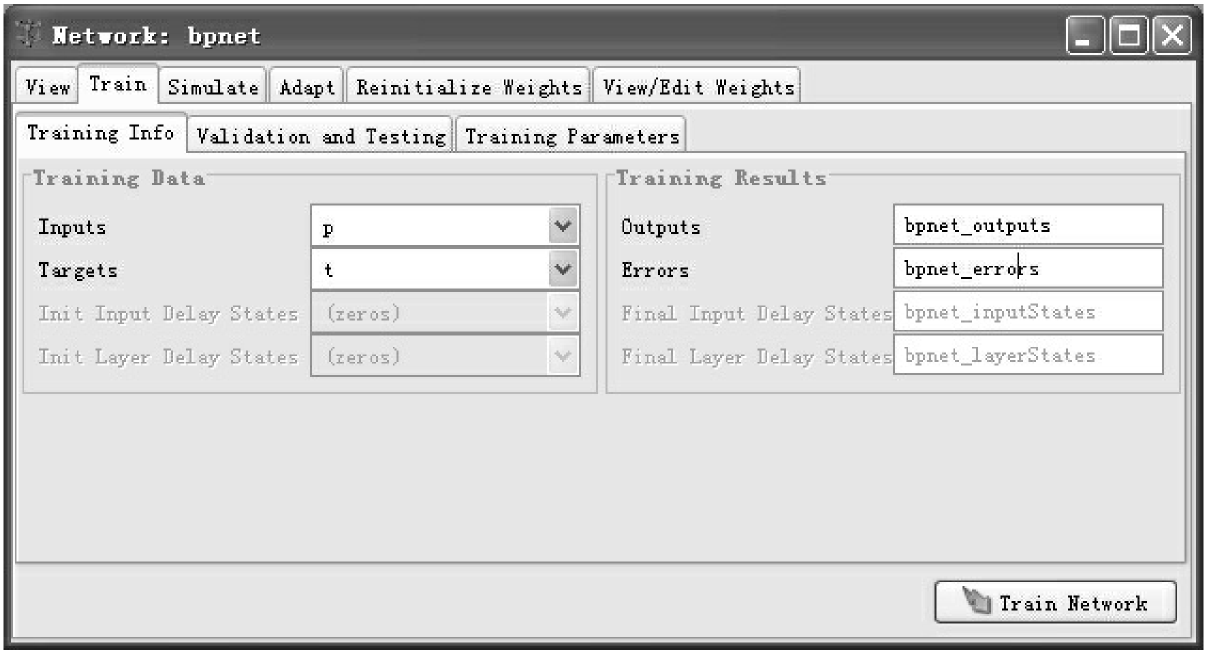
单击“Close”按钮关闭“Create Network or Data”对话框，回到如图 14.24所示的 nntool主界面中。



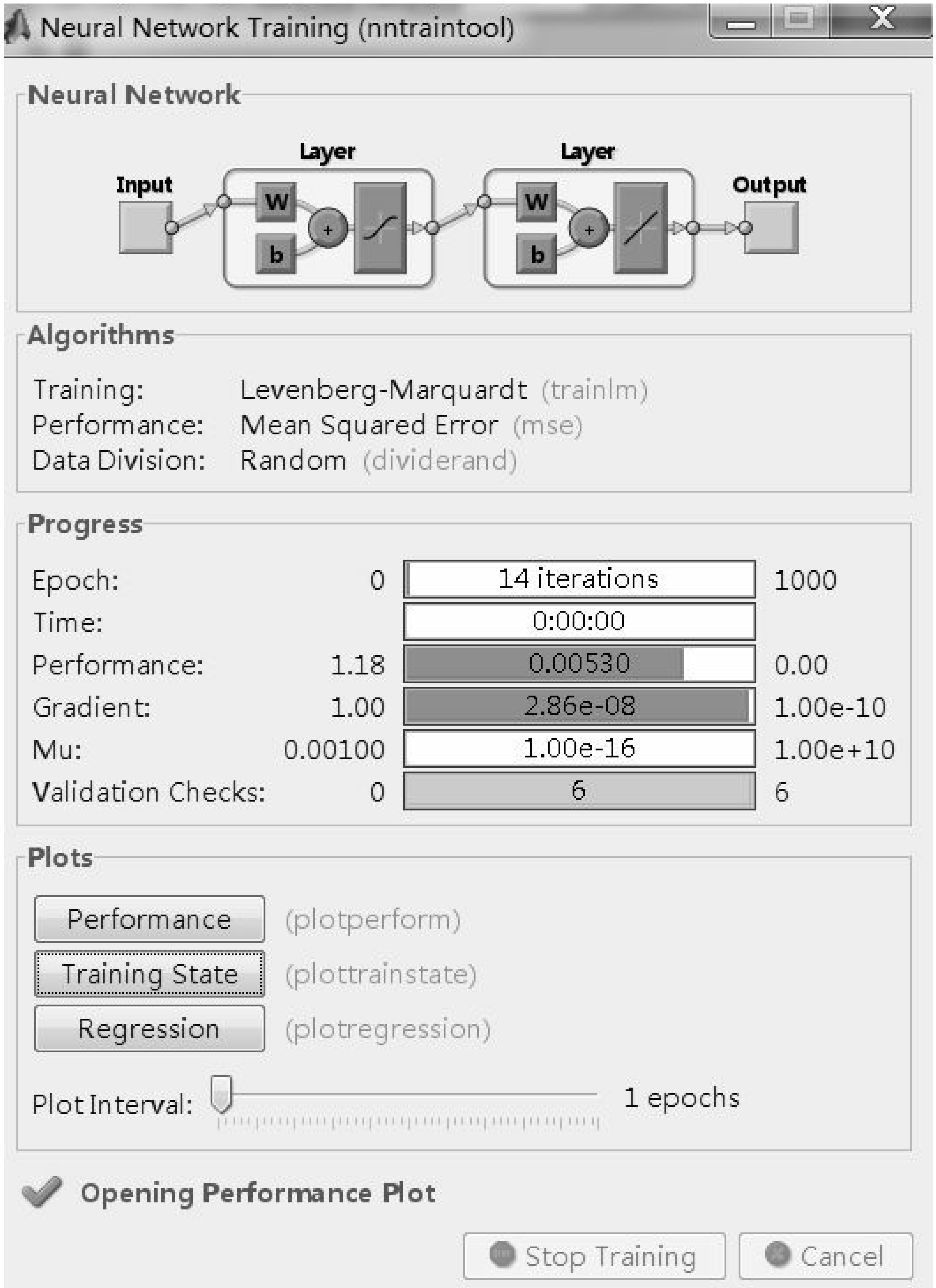
▲图14.24 nntool主界面

5．双击网络名称“bpnet”进入如图 14.25 所示的对话框，单击上方的“train”选项卡，在网络输入“Inputs”下拉选框中选择之前导入的变量p，Targets目标输出中选择，其他取默认设置。

单击“Train Network”按钮开始训练网络，得到图 14.26所示结果。


Network: bpnet
View
Train
Simulate
Adapt
Reinitialize Weights
View/Edit Weights
Training Info
Validation and Testing
Training Parameters
Training Data
Inputs
p
Targets
t
Init Input Delay States
(zeros)
Init Layer Delay States
(zeros)
Training Results
Outputs
bpnet_outputs
Errors
bpnet_errors
Final Input Delay States
bpnet_inputStates
Final Layer Delay States
bpnet_layerStates
Train Network

▲图14.25训练设置



▲图14.26训练结果图

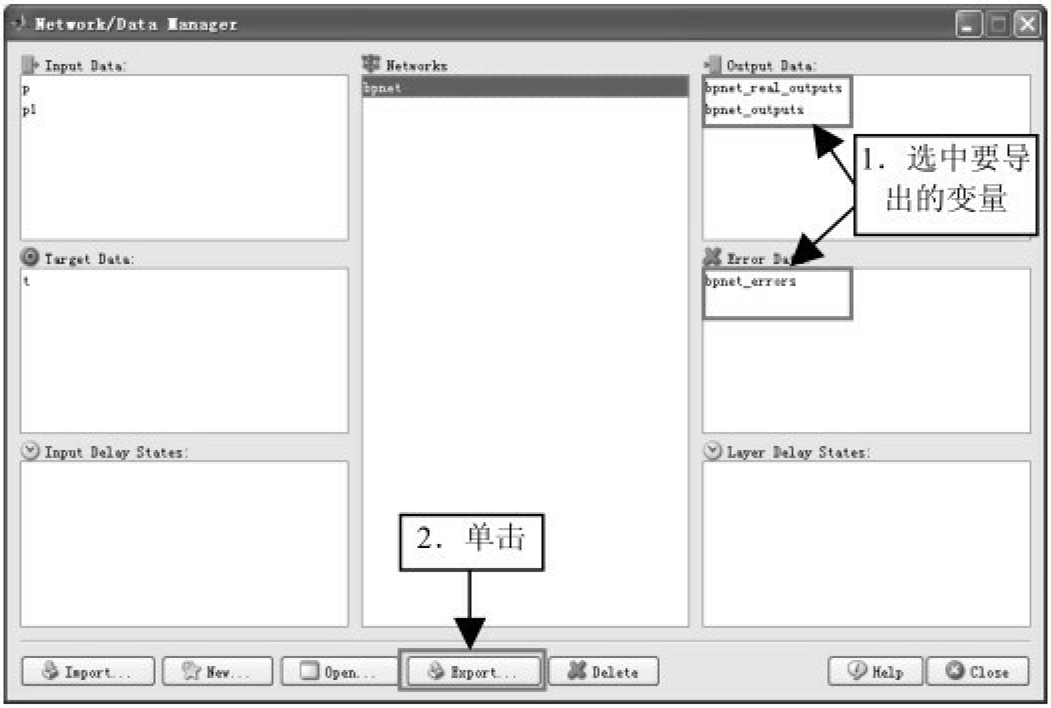
6．测试。单击图14.25上方的“Simulate”标签，输入p1，输出结果变

量bpnet_outputs改为bpnet_real_outputs，如图 14.27所示，单击“Simulate Network”按钮开始仿真测试。

7．选中要导出的变量，即bpnet_real_outputs, bpnet_outputs, bpnet_errors，单击“Export”按钮，将训练结果和测试结果导出到工作区，如图14.28所示。



▲图14.27仿真测试设置

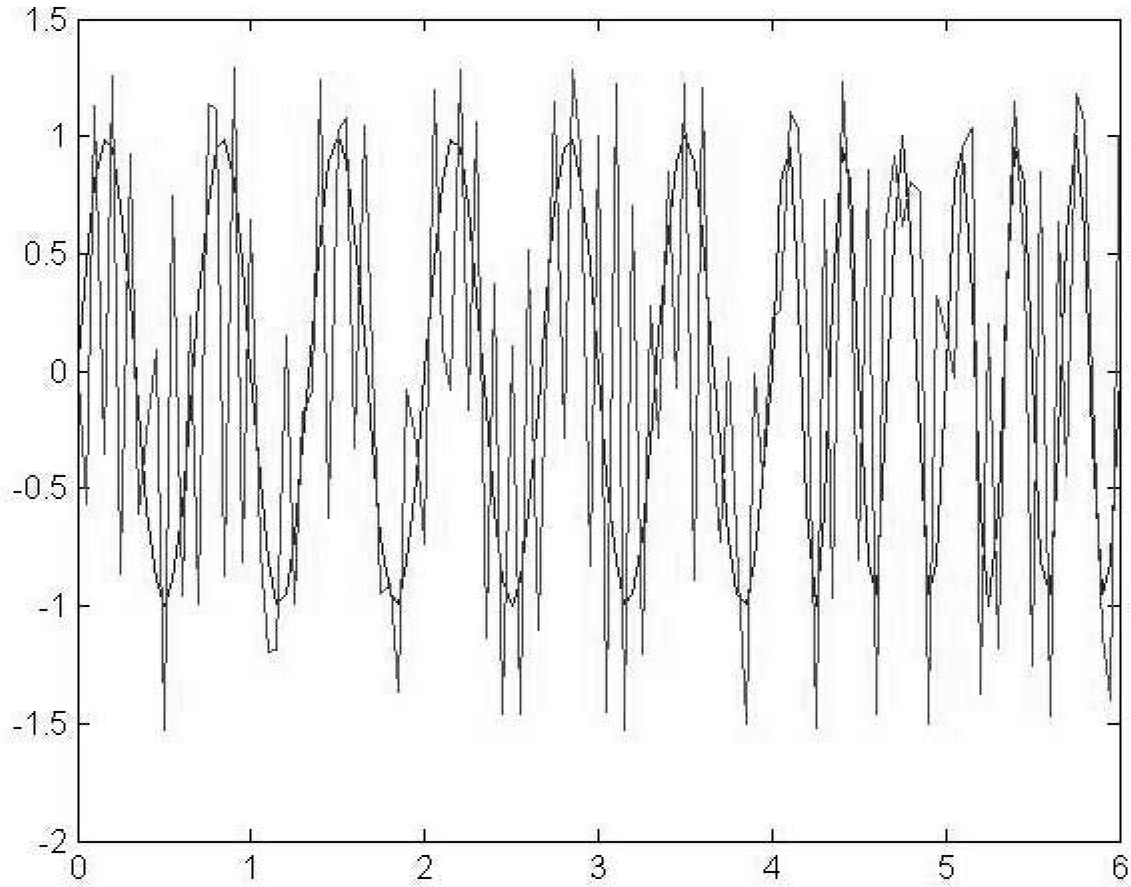


▲图14.28导出变量到工作区

8．使用下面的命令将训练目标t和实际测试输出进行可视化的比较。

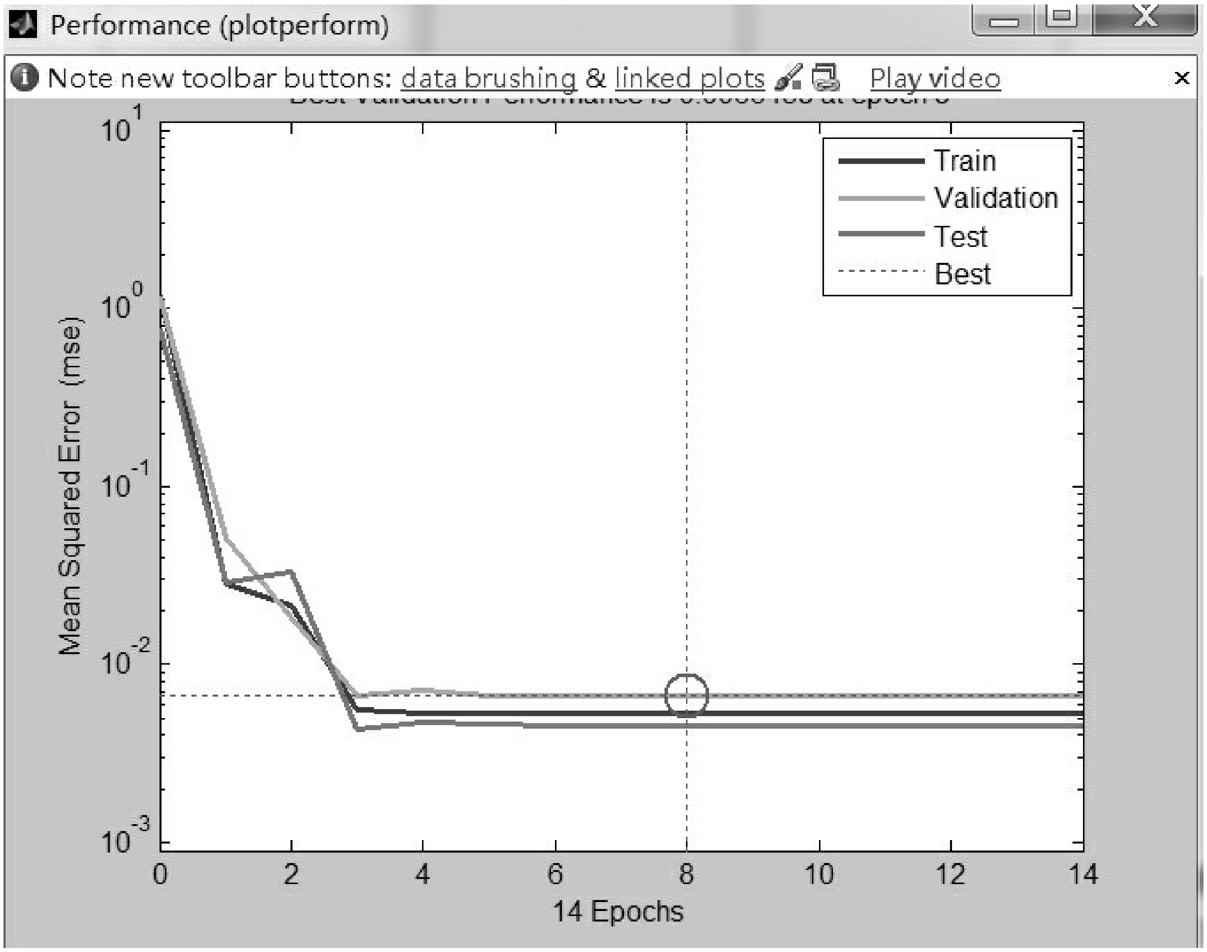
%训练目标t和预测结果bpnet_real_outputs图形

plot(time,t,time,bpnet_real_outputs);



▲图14.29 目标样本和预测结果对比图形

训练误差曲线如图14.30所示。



▲图14.30 训练误差曲线

14.4 Matlab神经网络工具箱

除了14.3节中介绍的可视化工具nntool之外，Matlab的神经网络工具箱还为我们提供了实现ANN 的众多函数接口。本节介绍从网络创建、初始化、权值调整，到训练和测试分析等主要功能的ANN工具箱函数。

14.4.1 网络的创建

Matlab2011中推荐使用feedforwardnet函数创建一个14.2节介绍的前馈网络，其功能等同于老版本中的newff函数，但在调用形式上更为简

洁，简化了用户的工作，该函数的原型为

Network = feedforwardnet(hiddenSizes,trainFcn)

该函数建立了网络对象并且初始化了网络权重和偏置，因此网络就可以进行训练了。

参数说明

hiddenSizes:一个指明了各层隐含层单元数目的行向量，默认值为10，表示网络含有一个单元数目为10的隐含层。

trainFcn：训练函数，即训练迭代中权值的更新方式，默认值为“trainlm”，是指L-M优化算法。

返回值

Network是一个经过初始化的网络结构。

建立好网络后，可以进一步对网络参数进相关设置。例如下面的命令可以设置网络中相应隐藏层的传递函数。

```
net.layers{1}.transferFcn = 'tansig';
net.layers{2}.transferFcn = 'logsig';
```

而下面的命令可以设置网络训练的迭代次数和终止误差判据。

```
net.trainParam.epochs = 1000;
net.trainParam.goal = 1e-5;
```

此外，也可以通过如下方式查看网络的相关信息。

```
net.inputs{1} %查看输入层信息
net.layers{1}, net.layers{2} %查看隐含层信息
net.biases{1} %查看偏置信息
net.outputs{2} %查看输出层信息
net.IW{1,1}, net.IW{2,1} %查看输入层权值
net.b{1} %查看偏置项值
net.LW{2,1} %查看隐含层权值
```

14.4.2 网络初始化

14.2.5 小节曾提到为了避免搜索陷入局部极小值，可以尝试不同初始位置的多次训练。ANN工具箱中的Init函数用于对网络进行重新的初始化。函数原型如下。

net = init(net)

其中，输入参数net是建立好的网络结构，输出net是经过重新初始化的网络结构，该函数根据 net.initFcn 中指定的更新函数更新网络的权值和偏置等信息，默认采用“initlay”，即根据net.layers{i}.initFcn初始化第i层网络的权值和偏置信息。

反向传播网络（Backpropagation networks中 net.layers{i}.initFcn默认设置为'initnw'，即使用Nguyen-Widrow方法进行初始化。其他类型网络的各层初始化方法可能相同。一种比较常用的初始化函数是rands，即采用−1～1的随机值进行初始化。

[例14.4]对使用feedforwardnet建立的网络进行重新初始化，并重新训练。

```
[x,t] = simplefit_dataset;
net = feedforwardnet(10);
net = train(net,x,t);
net2 = init(net);
net2 = train(net2,x,t);
```

14.4.3 网络训练

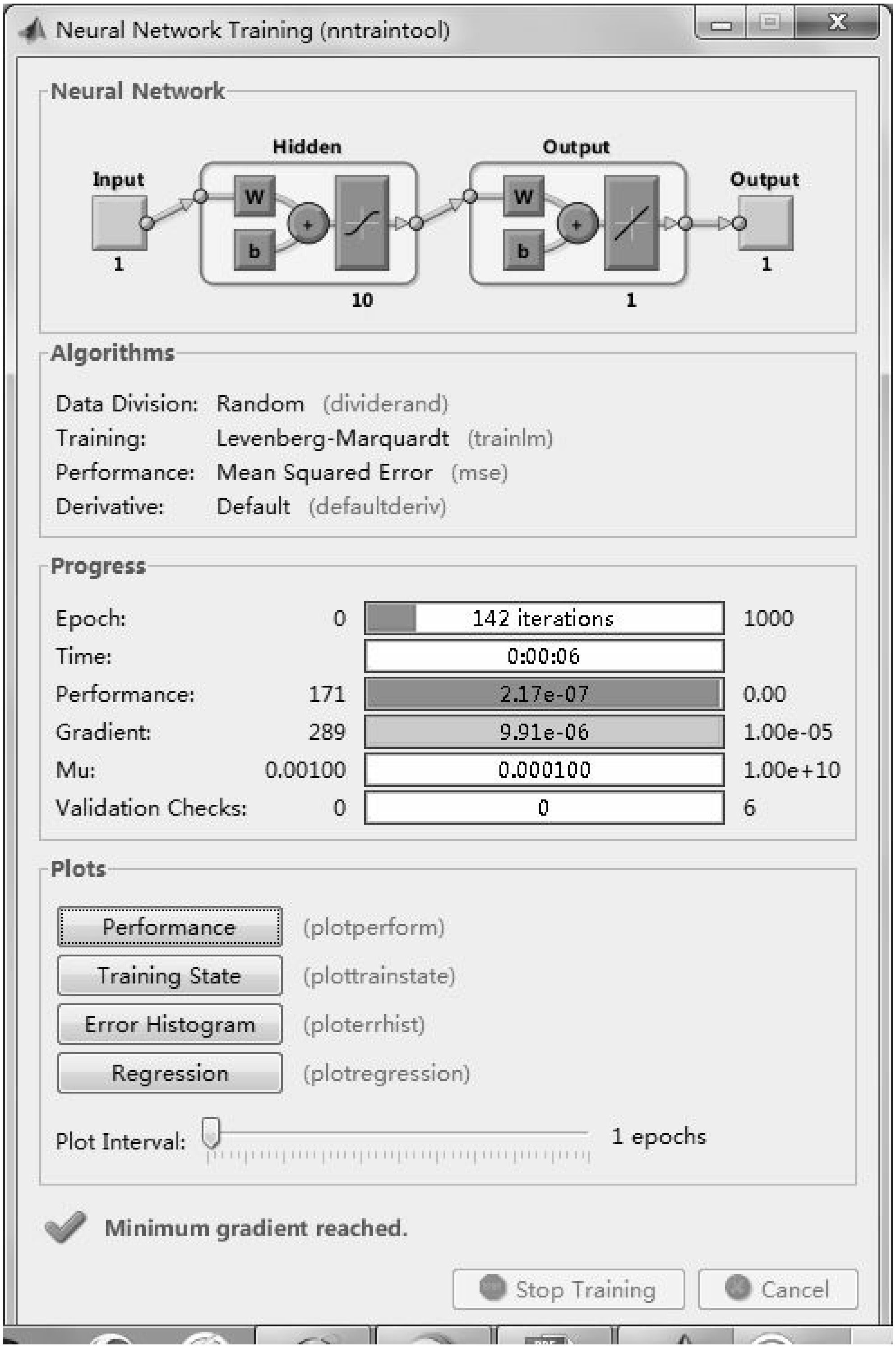
Train函数用于网络的训练。其常用调用形式如下。

[NET,TR] = train(NET,X,T)

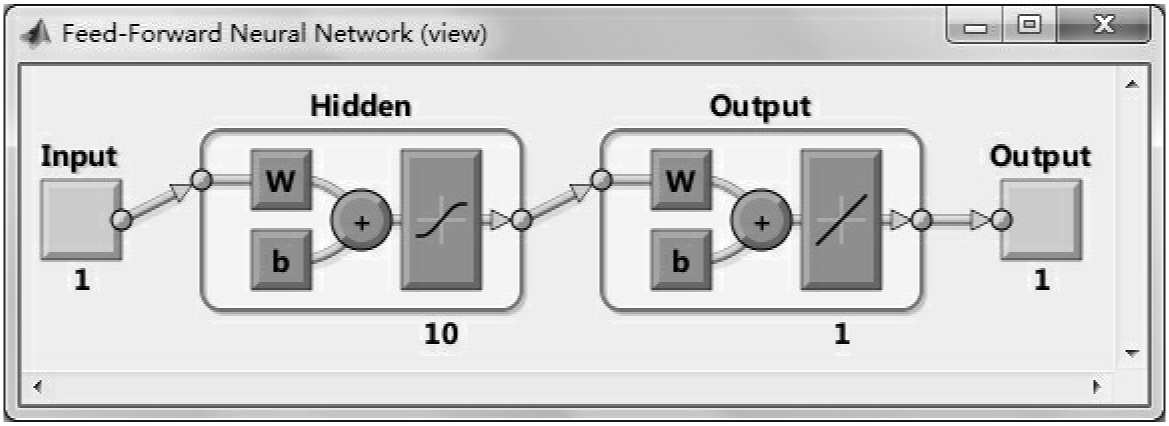
其中：输入参数 NET 是尚未训练的网络结构；X 是训练样本数据；T 是训练样本的类标签。函数返回一个训练好的网络结构NET以及

一个训练机构机构TR。默认情况下训练时会出现一个如图14.31所示的训练进程界面。

对于训练好的网络，可以通过view(net)命令来查看网络结构，效果如图14.32所示。


Neural Network Training (nntraintool)
Neural Network
Hidden
Output
Input
Output
1
10
1
1
Algorithms
Data Division: Random (dividerand)
Training: Levenberg-Marquardt (trainlm)
Performance: Mean Squared Error (mse)
Derivative: Default (defaultderiv)
Progress
Epoch: 0 142 iterations 1000
Time: 0:00:06
Performance: 171 2.17e-07 0.00
Gradient: 289 9.91e-06 1.00e-05
Mu: 0.00100 0.000100 1.00e+10
Validation Checks: 0 0 6
Plots
Performance (plotperform)
Training State (plottrainstate)
Error Histogram (ploterrhist)
Regression (plotregression)
Plot Interval: 1 epochs
Minimum gradient reached.
Stop Training
Cancel

▲图14.31训练进程



▲图14.32网络结构

14.4.4 网络仿真测试

sim函数使用训练好的网络进行仿真，即测试。其常用调用形式为

Y = sim(net, X);

其中：net为训练好的网络结构；X是神经网络的输入；返回值Y为对应于X的网络输出。

上述sim函数的调用方法也等同于下面的调用方式。

Y = net(X);

14.4.5 网络性能分析

Performance函数用于度量网络的分类性能，即分类误差。其常用调用形式如下。

perf = perform(net,y,t)

其中：输入参数 net 是训练好的网络结构；y 是网络的实际输出；t 是网络的目标输出。输出perf给出了实际输出与目标输出之间的误差度量。

下面通过一个简单的实例来说明上述函数的用法。

[例14.5]建立一个前馈神经网络并针对simplefit数据集进行训练，测

试其性能。

```
[x,t] = simplefit_dataset; %载入数据集
net = feedforwardnet(10) %创建网络
net = train(net,x,t); %训练网络
view(net) %查看网络结构
y = net(x); % 测试网络
perf = perform(net,y,t); % 网络性能
```

以上是神经网络训练和识别中几个最为常用的函数，Matlab的ANN工具箱中与神经网络相关的函数还有很多。读者可以在 Matlab 命令行窗口中键入“help nnet+Enter”命令，从而获得关于Matlab中神经网络工具箱中更多函数的帮助索引信息。

第15章 支持向量机

支持向量机（Support Vector Machine, SVM）是在统计学习理论的基础上发展起来的新一代学习算法，它在文本分类、手写识别、图像分类、生物信息学等领域中获得较好的应用。相比于容易过渡拟合训练样本的人工神经网络，支持向量机对于未见过的测试样本具有更好的推广能力（generalization ability）。

本章的知识和技术热点

- SVM的理论基础
- 核函数
- SVM推广到多类问题的 3种策略

本章的典型案例分析

基于PCA和SVM的人脸识别系统

15.1 支持向量机的分类思想

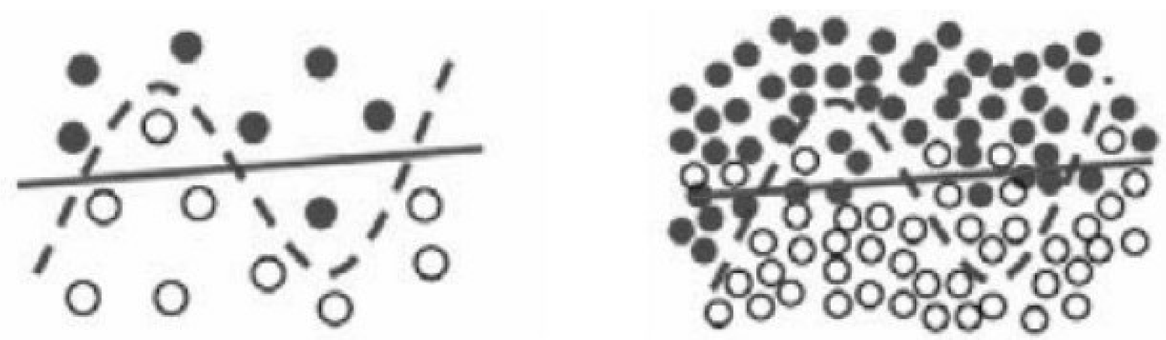
传统模式识别技术只考虑分类器对训练样本的拟合情况，以最小化训练集上的分类错误为目标，通过为训练过程提供充足的训练样本来试图提高分类器在未见过的测试集上的识别率。然而，对于少量的训练样本集合来说，我们不能保证一个很好地分类了训练样本的分类器也能够很好地分类测试样本。在缺乏代表性的小训练集情况下，一味地降低训练集上的分类错误就会导致过渡拟合。

支持向量机以结构化风险最小化为原则，即兼顾训练误差（经验风险）与测试误差（期望风险）的最小化，具体体现在分类模型的选择和模型参数的选择上。

15.1.1 分类模型的选择

要分类如图15.1（a）中所示的2类样本，我们看到图中的曲线可以将（a）中的训练样本全部分类正确，而直线则会错分2个训练样本；然而，对于（b）中的大量测试样本，简单的直线模型却取得了更好的识别结果。应该选择什么样的分类模型呢？

图 15.1 中复杂的曲线模型过渡拟合了训练样本，因而在分类测试样本时效果并不理想。在第13.1.5小节，我们了解到通过控制分类模型的复杂性可以防止过度拟合，因此SVM更偏爱解释数据的简单模型——二维空间中的直线，三维空间中的平面和更高维空间中的超平面。

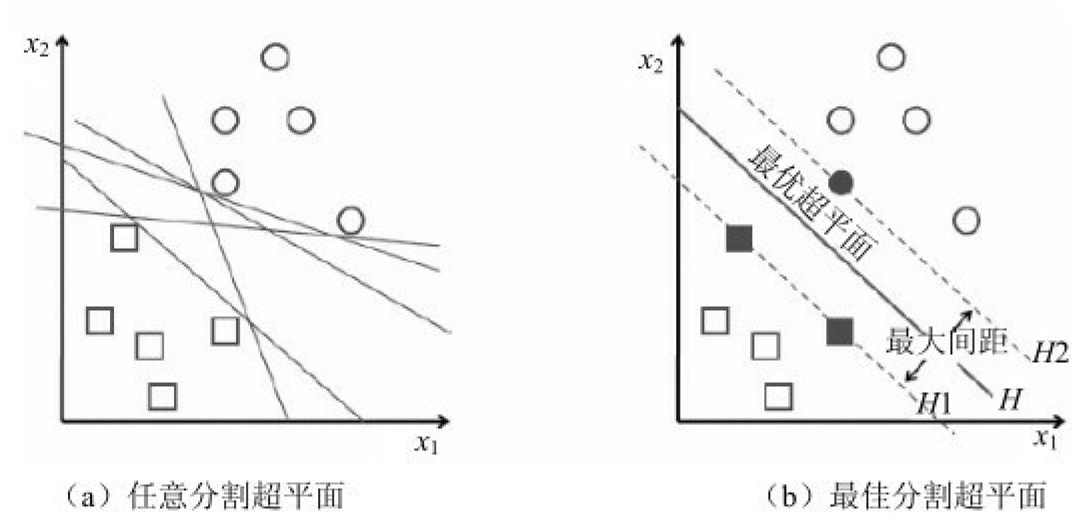


（a）训练样本上的 2 种分类模型　（b）测试样本上的 2 种分类模型

▲图15.1 分类模型的选择

15.1.2 模型参数的选择

图15.2所示为二维空间中的两类样本，可以采用图15.2（a）中的任意直线将它们分开。哪条直线才是最优的选择呢？



▲图15.2 分割超平面

直观上，距离训练样本太近的分类线将对噪声比较敏感，且对训练样本之外的数据不太可能归纳得很好；而远离所有训练样本的分类线将可能具有较好的归纳能力。设H为分类线，H1、H2分别为过各类中离分类线最近的样本且平行于分类线的直线，则H1与H2之间的距离叫做分类间隔（又称为余地，margin）。所谓最优分类线就是要求分类线不但能将两类正确分开（训练错误率为0），而且使分类间隔最大，如图15.2（b）所示。分类线的方程为$w^Tx + b = 0$。

图 15.2 只是在二维情况下的特例——最优分类线，在三维空间中则是具有最大间隔的平面，更为一般的情况是最优分类超平面。实际上，SVM 正是从线性可分情况下的最优分类面发展而来的，其主要思想就是寻找能够成功分开两类样本并且具有最大分类间隔的最优分类超平面。

寻找最优分类面的算法最终将转化成为一个二次型寻优问题，从理论上讲，得到的将是全局最优点，解决了在神经网络方法中无法避免的局部极值问题。

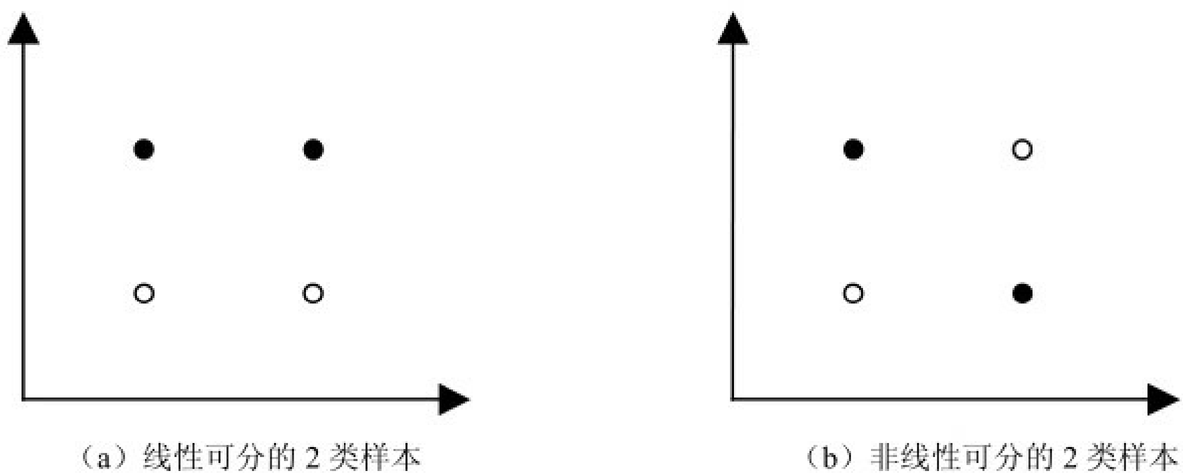
15.2 支持向量机的理论基础

本节主要介绍SVM的理论基础和实现原理，将分别阐述线性可分、非线性可分以及需要核函数映射这3种情况下的SVM。最后还将学习如何将SVM推广至多类问题。

15.2.1 线性可分情况下的SVM

如果用一个线性函数（如二维空间中的直线，三维空间中的平面以及更高维数空间中的超平面）可以将两类样本完全分开，就称这些样本是线性可分（**linearly separable**）的。反之，如果找不到一个线性函数能够将两类样本分开，则称这些样本是非线性可分的。

一个简单的线性可分与非线性可分的例子如图15.3所示。



▲图15.3 线性可分与非线性可分

已知一个线性可分的数据集$\{(\vec{x}_1,y_1),(\vec{x}_2,y_2),\ldots,(\vec{x}_N,y_N)\}$，样本特征向量$\vec{x}\in r^D$，即$\vec{x}$是D维实数空间中的向量；类标签 $y\in\{-1, +1\}$，即只有 2类样本，此时通常称类标签为+1的样本为正例，称类标签为–1的样本为反例。

现在要对这2类样本进行分类。我们的目标就是寻找最优分割超平面，即根据训练样本确定最大分类间隔的分割超平面，设最优超平面方程为$\vec{w}^{T}\vec{x}$+b=0，根据点到平面的距离公式：样本$\vec{x}$与最佳超平面（$\vec{w}$，b）之间的距离为$\frac{|\vec{w}^T\vec{x}+b|}{||\vec{w}||}$，注意通过等比例地缩放权矢量$\vec{w}$和偏差项b最佳超平面存在着许多解，我们对超平面进行规范化，选择使得距超平面最近的样本$\vec{x}_k$满足|$\vec{w}^{T}\vec{x}_k$+b|=1的$\vec{w}$和b，即得到规范化超平面。

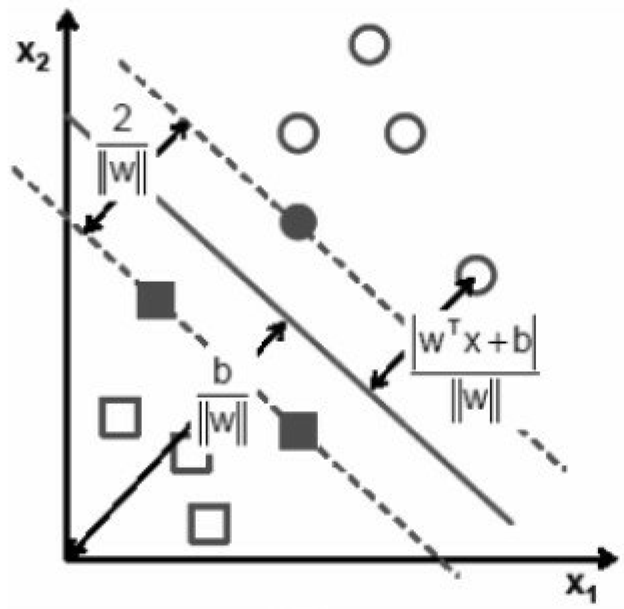
此时从最近样本到Z边缘的距离为

$$\frac{|\vec{w}^T\vec{x}_k+b|}{||\vec{w}||}=\frac{1}{||\vec{w}||} \qquad (15\text{-}1)$$

且分类间隔（余地）变为

$$m=\frac{2}{||\vec{w}||} \qquad (15\text{-}2)$$

如图15.4所示。



▲图15.4 最佳分割超平面的分类间隔

至此，问题逐渐明朗化，我们的目标是寻找使得式（15-2）最大化的法向量$\vec{w}$，之后将$\vec{w}$代入关系式$|\vec{w}^T\vec{x}_k+b|=1$，即可得到b。

最大化式（15-2）等价于最小化。

$$J(\vec{w})=\frac{1}{2}\|\vec{w}\|^2 \quad （15\text{-}3）$$

除此之外，还有以下的约束条件。

$$y_i(\vec{w}^T\vec{x}_i+b)\geq 1,\quad \forall i\in\{1,2,\cdots,N\} \quad （15\text{-}4）$$

这是因为距离超平面最近的样本点$\vec{x}_k$满足$|\vec{w}^T\vec{x}_k+b|=1$，而其他样本点$\vec{x}_i$距离超平面的距离$d(\vec{x}_i)$要大于等于$d(\vec{x}_k)$，因此有

$$|\vec{w}^T\vec{x}_i+b|\geqslant 1 \quad （15\text{-}5）$$

具体地说，我们设定正例所在的一侧超平面的正方向，则对于正例（对应类标签 y_i为+1 的样本$\vec{x}_i$）有

$$(\vec{w}^T\vec{x}_i + b) \geqslant 1 \quad (15\text{-}6)$$

而对于反例（对应类标签y_i为-1的样本$\vec{x}_i$）有

$$(\vec{w}^T\vec{x}_i + b) \leqslant -1 \quad (15\text{-}7)$$

在式（15-6）和式（15-7）的两端分别乘以对应其$\vec{x}_i$的类标签y_i，由于对应式（15-6）和式（15-7）的y_i分别为+1和-1，因此得到式（15-4）中统一形式的表达式。

注意到式（15-3）中的目标函数J($\vec{w}$)是二次函数，意味着只存在一个全局最小值，因此我们不必再像在神经网络的优化过程中那样担心搜索陷入局部极小值。现在要做的就是在式（15-4）的约束条件下找到能够最小化式（15-3）的超平面方向量$\vec{w}$。这是一个典型的条件极值问题，可以使用我们在高等数学中学习过的拉格朗日乘数法求解。

通过对式（15-4）中的每一个约束条件乘上一个拉格朗日乘数α_i，然后带入式（15-3）中，可将此条件极值问题转化为下面的不受约束的优化问题，即关于$\vec{w}$，b和α_i（i=1,2,...,N）来最小化L。

$$L(\vec{w},b,\alpha) = \frac{1}{2}\|\vec{w}\|^2 - \sum_{i=1}^{N}\alpha_i[y_i(\vec{w}^T\vec{x}_i + b) - 1], \quad \alpha_i \geqslant 0 \quad (15\text{-}8)$$

求L对$\vec{w}$和b的偏导数，并令其等于零。

$$\frac{\partial L(\vec{w},b,\alpha)}{\partial \vec{w}}=0 \Rightarrow \vec{w}=\sum_{i=1}^{N}\alpha_i y_i \vec{x}_i \tag{15-9}$$

$$\frac{\partial L(\vec{w},b,\alpha)}{\partial b}=0 \Rightarrow \sum_{i=1}^{N}\alpha_i y_i =0 \tag{15-10}$$

展开式（15-8），得

$$L(\vec{w},b,\alpha)=\frac{1}{2}\vec{w}^T\vec{w}-\sum_{i=1}^{N}\alpha_i y_i \vec{w}^T x_i - b\sum_{i=1}^{N}\alpha_i y_i+\sum_{i=1}^{N}\alpha_i \tag{15-11}$$

再将式（15-9）和式（15-10）代入式（15-11），得

$$\begin{aligned}L(\vec{w},b,\alpha)&=\frac{1}{2}\vec{w}^T(\sum_{i=1}^{N}\alpha_i y_i \vec{x}_i)-\vec{w}^T\sum_{i=1}^{N}\alpha_i y_i \vec{x}_i-0+\sum_{i=1}^{N}\alpha_i\\&=-\frac{1}{2}(\sum_{i=1}^{N}\alpha_i y_i \vec{w}^T\vec{x}_i)+\sum_{i=1}^{N}\alpha_i\\&=-\frac{1}{2}\sum_{i=1}^{N}\alpha_i y_i(\sum_{j=1}^{N}\alpha_j y_j \vec{x}_j)^T\vec{x}_i+\sum_{i=1}^{N}\alpha_i\\&=-\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j \vec{x}_i^T\vec{x}_j+\sum_{i=1}^{N}\alpha_i\end{aligned}$$

上式与$\vec{w}$、b无关，仅为α_i的函数，记为

$$L(\alpha)=-\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j \vec{x}_i^T\vec{x}_j+\sum_{i=1}^{N}\alpha_i \tag{15-12}$$

此时的约束条件为$\alpha_i \geq 0$，并且

$$\sum_{i=1}^{N}\alpha_i y_i=0。 \tag{15-13}$$

这是一个拉格朗日对偶问题，而该对偶问题是一个关于α的凸二次

规划问题，可借助一些标准的优化技术求解，这里不再详细讨论。

注意

原始问题的复杂度由维度决定（$\vec{w}$是平面法向量，对每维有1个系数），然而对偶问题由训练数据的数目决定（每个样本存在一个拉格朗日乘数α i）；此外，在式（15-12）中，训练数据只作为点积的形式出现。

在解得α之后，最大余地分割超平面的参数$\vec{w}$ 和b便可由对偶问题的解α来确定。

$$\vec{w}=\sum_{i=1}^{N}\alpha_i y_i x_i \qquad (15\text{-}14)$$

在样本线性可分的情况下，由于有关系式|$\vec{w}^{\,T}\vec{x}_k$+b|=1，其中$\vec{x}_k$是任意一个距离最优分类超平面最近的向量，即可使|$\vec{w}^{\,T}\vec{x}_k$+b|取到最小值的$\vec{x}_k$之一，故可将$\vec{w}$和$\vec{x}_k$代入上式，从而求出b。

$$b=1-\min_{y_i=+1}(\vec{w}\cdot\vec{x}_i) \quad 或 \quad b=-1-\max_{y_i=-1}(\vec{w}\cdot\vec{x}_i)$$

更一般的情况下（包括15.2.2小节的非线性可分的情况），由于两类样本中与分割超平面(w, b)的最近距离不再一定是1且可能不同，因而b为

$$b=-\frac{1}{2}(\min_{y_i=+1}(w\cdot x_i)+\max_{y_i=-1}(w\cdot x_i)) \qquad (15\text{-}15)$$

式（15-15）包含了线性可分时的情况。

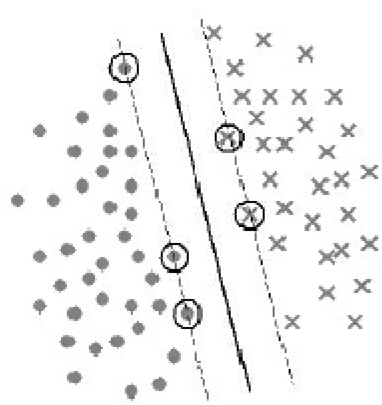
根据优化解的性质（Karush－Kuhn－Tucker conditions），解α必须满足

$$\alpha_i[y_i(\vec{w}^T\vec{x}_i+b)-1]=0,\quad \forall i=1\ldots N \tag{15-16}$$

因此对于每个样本，必满足α_i=0或$y_i(\vec{w}^T\vec{x}_i+b)-1=0$。从而对那些满足$y_i(\vec{w}^T\vec{x}_i+b)-1\neq 0$的样本$\vec{x}_i$对应的$\alpha_i$，必有$\alpha_i$=0；而只有那些满足$y_i(\vec{w}^T\vec{x}_i+b)-1=0$的样本$\vec{x}_i$对应的$\alpha_i$，才能有$\alpha_i$>0。这样，$\alpha_i$>0只对应那些最接近超平面（$y_i(\vec{w}^T\vec{x}_i+b)-1=0$）的点$\vec{x}_i$，这些点被称为支持向量，如图15.5所示。

注意

在式（15-14）中，所有α_i=0的样本$\vec{x}_i$对于求和没有影响，只有支持向量（$\alpha_i\neq0$的样本$\vec{x}_i$）对最优分割超平面的定义有贡献。因此，完整的样本集合可以只使用支持向量来代替，将会得到相同的最优分割超平面。



▲图15.5 支持向量。两类样本的支持向量分别用“零”圈出

求出上述各个系数α、$\vec{w}$、b对应的最优解α^*、$\vec{w}^*$、b^*后，得到如下的最优分类函数。

$$h(\vec{x})=\mathrm{sgn}((\vec{w}^*\vec{x})+b^*)=\mathrm{sgn}(\sum_{i=1}^{n}\alpha_i^* y_i(\vec{x}_i\cdot\vec{x})+b^*) \tag{15-17}$$

这里向量$\vec{x}$是待分类的测试样本，向量$\vec{x}_i$（i=1,2,…,N）是全部N个训练样本。注意在式（15-17）中测试样本$\vec{x}$与训练样本$\vec{x}_i$也是以点积的形式出现。

15.2.2 非线性可分情况下的C-SVM

1．约束条件

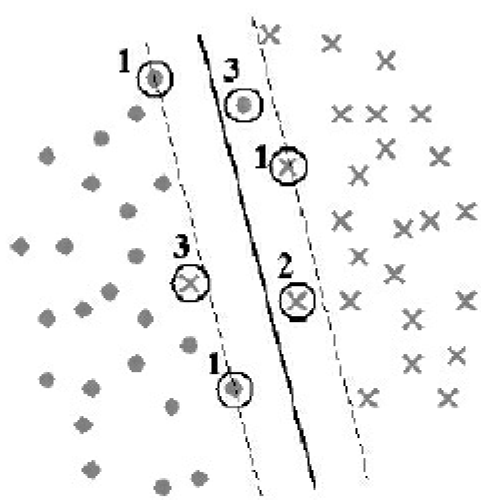
为处理样本非线性可分的情况，我们放宽约束，引入松弛变量ε_i>0，此时约束条件变为

$$y_i(\vec{w}\cdot\vec{x}_i+b)\geqslant 1-\varepsilon_i,\quad \varepsilon_i\geqslant 0\ (i=1,2,\cdots,N) \tag{15-18}$$

即

$$\begin{cases}\vec{w}\cdot\vec{x}_i+b\geqslant +1-\varepsilon_i, & y_i=+1(\vec{x}_i\text{为正例})\\ \vec{w}\cdot\vec{x}_i+b\leqslant -1+\varepsilon_i, & y_i=+1(\vec{x}_i\text{为反例})\end{cases}$$

图15.6可帮助我们理解ε_i的意义，具体可分为以下3种情况来考虑。



▲图15.6 非线性可分情况下的最佳分割超平面。

标号1：分类间隔支持向量，对应$\varepsilon_i=0$；

标号2：非分类间隔支持向量（分类间隔中），对应$0<\varepsilon_i<1$；

标号3：非分类隔支持向量$\varepsilon_i>1$。

● 当$\varepsilon_i=0$，约束条件退化为线性可分时的情况——$y_i(\vec{w}^T\vec{x}_i+b)\geq 1$，这对应于分类间隔（余地）以外且被正确分类的那些样本，即图中左侧虚线以左（包括在左侧虚线上）的所有“•”形样本点以及位于右侧虚线以右的（包括在右侧虚线上）的所有“x”形样本点。

● 当$0<\varepsilon_i<1$， $yi(\vec{w}^T\vec{x}_i+b)$是一个0～1之间的数，小于1意味着我们对于约束条件放宽到允许样本落在分类间隔之内，大于 0 则说明样本仍可被分割超平面正确分类，对应于图中标号为 2的样本。

● 最终当$\varepsilon_i>1$，$y_i(\vec{w}^T\vec{x}_i+b)<0$，此时约束条件已放宽到可以允许有分类错误的样本，如图中的第3类样本。具体地说，图中标号为3的“•”形样本的$1<\varepsilon_i<2$，而标号为3的“x”形样本的$\varepsilon_i>2$。

图15.6中标号为1、2、3的均为在线性不可分情况下的支持向量。由于在这种情况下允许样本落入分类间隔之内，我们常把这个分类间隔叫做软间隔（soft margin）。

2．目标函数

利用一个附加错误代价系数C后，目标函数变为

$$f(\vec{w},b,\varepsilon)=\frac{1}{2}\|\vec{w}^2\|+C\sum_{i=1}^{N}\varepsilon_i \tag{15-19}$$

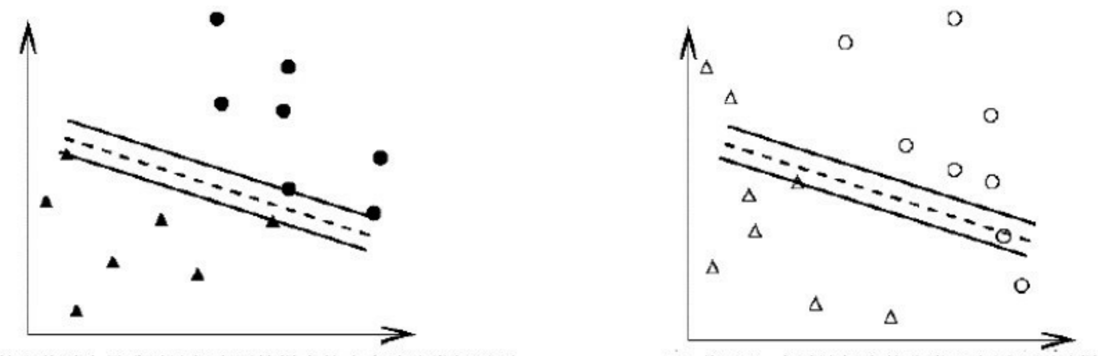
我们的目标是在式（15-18）的约束下，最小化目标函数式（15-19）。最小化目标函数的第 1项也就等同于最大化分类间隔，这在介绍线性可分情况时已经阐述过了，而目标函数的第2项是分类造成的错误代价，只有对应于$\varepsilon_i>0$ 的那些“错误”样本才会产生代价（这里所说

的“错误”并不仅仅指被错误分类的标号为3的样本，也包括那些空白间隔之内的标号为2的样本）。事实上最小化此目标函数体现了最大分类间隔（最小化式（15-19）中的第 1 项）与最小化训练错误（最小化式（15-19）中的第2项）之间的权衡。

直观上我们自然希望“错误”样本越少越好，然而不要忘记这里的错误是训练错误，如果我们一味追求最小化训练错误代价就可能导致得到一个小余地的超平面，这无疑会影响分类器的推广能力，在对测试样本分类时就很难得到满意的结果，这也属于一种过渡拟合。通过调整代价系数 C的值可以实现两者之间的权衡，找到一个最佳的C使得分类超平面兼顾训练错误和推广能力。

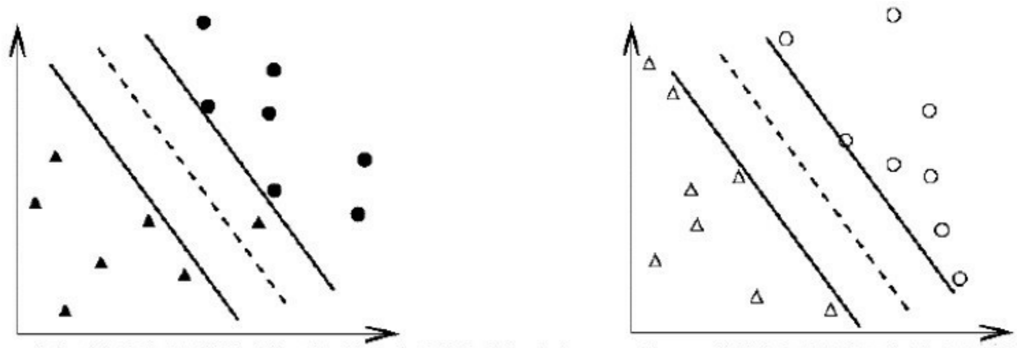
不同的C值对于分类的影响如图15.7所示。图15.7（a）中的情况对应一个相对较大的C值，此时每错分一个样本i（$\varepsilon_i>0$）都会使式（15-19）中的第2项增大很多，第2项成为影响式（15-19）的主要因素，因此最小化式（15-19）的结果是尽可能少地错分训练样本以使得第 2 项尽可能小，为此可以适当牺牲第1项（使第1项大一些，即使分类间隔小一些），于是导致了图15.7（a）中一个较小间隔但没有错分训练样本的分类超平面；图15.7（b）图展示了将图15.7（a）中得到的分类超平面应用于测试样本的效果，可以看出由于分割超平面间隔较小，分类器的推广能力不强，对于测试样本的分类不够理想。

如果在训练过程中选择一个适当小一些的C值，此时最小化式（15-19）将兼顾训练错误与分类间隔。如图15.7（c）所示，虽然有一个训练样本被错分，但得到了一个较大分类间隔的超平面；图15.7（d）展示了将图15.7（c）中得到的分类超平面应用于测试数据时的情形，我们看到由于分类间隔较大，分类器具有良好的推广能力，从而很好地分类了测试样本。



（a）C 的取值过大导致过渡拟合训练样本的小余地分类超平面　（b）将（a）中过渡拟合的分类器应用于测试样本

▲图15.7 惩罚项参数C的不同取值对于分类器性能的影响



（c）取值合适的 C：允许适当地错分训练样本从而得到较大余地的分类超平面　（d）将（c）的较大余地分类器应用于测试样本

▲图15.7 惩罚项参数C的不同取值对于分类器性能的影响（续）

由此我们看到了选择合适的错误代价系数C的重要性，在15.4.3小节关于SVM综合案例的讨论中将提供一种切实可行的方法来选择 C 的取值。而正因为在这种处理非线性可分问题的方法中引入了错误代价系数C，这种支持向量机常被称为C-SVM。

3．优化求解

类似于线性可分情况下的推导，最终得到下面的对偶问题。

在如下的约束条件下，

$$\sum_{i=1}^{N}\alpha_i y_i = 0,\quad 0 \leqslant \alpha_i \leqslant C, i = 1, 2, \cdots, N \tag{15-20}$$

最大化L(α)，有

$$L(\alpha)=\sum_{i=1}^{N}\alpha_i-\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j(\vec{x}_i\cdot\vec{x}_j) \quad (15\text{-}21)$$

同样在利用二次规划技术解得最优的α的值α^*之后，可以计算出w^*和b^*的值，最终的决策函数与式（15-17）相同。

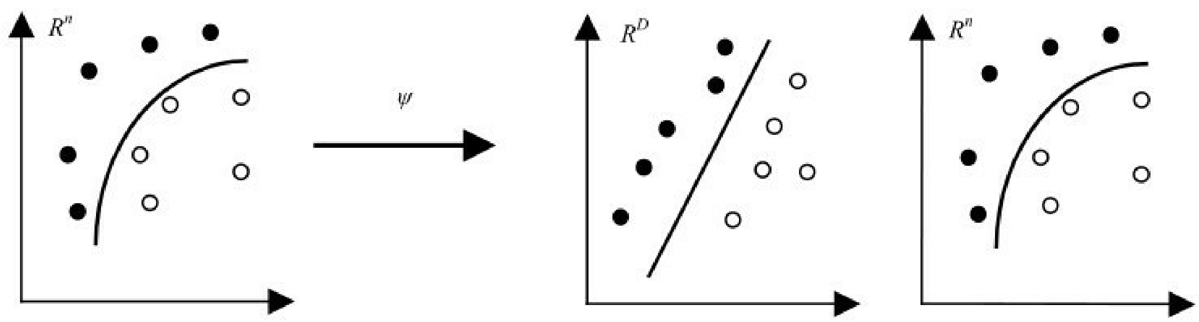
15.2.3 需要核函数映射情况下的SVM

线性分类器的分类性能毕竟有限，而对于非线性问题一味放宽约束条件只能导致大量样本的错分，这时可以通过非线性变换将其转化为某个高维空间中的线性问题，在变换空间求得最佳分类超平面。

1．非线性映射

如图15.8所示，图（a）中给出了是在n维空间中非线性可分的2类样本（限于图的表现能力，只画出了二维），通过一个非线性映射ψ：$r^n\rightarrow r^D$将样本映射到更高维的特征空间r^D（限于图的表现能力，只画出了二维），映射后样本$\psi(\vec{x}_i)$（i=1,2,…,N）在新的特征空间r^D中线性可分，经训练可得到一个图15.8（b）中所示的D维的分割超平面。而再将此D维分割超平面映射回r^n，该超平面可能就对应于原n维中的一条能够完全分开2类样本的超抛物面，如图15.8（c）所示。

图15.8所展示的只是一种比较理想的情况，实际中样本可能在映射到高维空间r^D后仍非线性可分，这时只需在r^D中采用15.2.2小节介绍的非线性可分情况下的方法训练SVM。还有一点要说明的是在分类时我们永远不需要将r^D中的分割超平面再映射回r^n当中，而是应让分类样本xG也经非线性变换ψ映射到空间r^D中，然后将$\psi(\vec{x})$送入r^D中的SVM分类器即可。



▲图15.8 使用非线性映射ψ将已知样本映射到（高维）特征空间RD

2．优化求解

类似于15.2.1小节中的推导，最终得到了下面的对偶问题。

在如下的约束条件下，有

$$\sum_{i=1}^{N}\alpha_i y_i = 0, \quad 0 \le \alpha_i \le C, i = 1, 2, \cdots, N \tag{15-22}$$

最大化目标函数为

$$L(\alpha) = \sum_{i=1}^{N}\alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i \alpha_j y_i y_j (\psi(\vec{x}_i)\cdot\psi(\vec{x}_j)) \tag{15-23}$$

此时，因为有 $\vec{w} = \sum_{i=1}^{N}\alpha_i y_i \psi(\vec{x}_i)$

故最终的决策（分类）函数为

$$h(\vec{x}) = \mathrm{sgn}(\vec{w}\cdot\psi(\vec{x}) + b) = \mathrm{sgn}(\sum_{i=1}^{N}\alpha_i y_i (\psi(\vec{x}_i)\cdot\psi(\vec{x})) + b), \quad \vec{w} \in R^D, b \in R \tag{15-24}$$

同样，由于对非支持向量而言其$\alpha_i = 0$，所以上式可以写成

$$h(x) = \mathrm{sgn}(\sum_{i \in SV}\alpha_i^{*} y_i (\psi(\vec{x}_i)\cdot\psi(\vec{x})) + b^{*}) \tag{15-25}$$

其中，SV表示支持向量（Support Vector）的集合。

注意

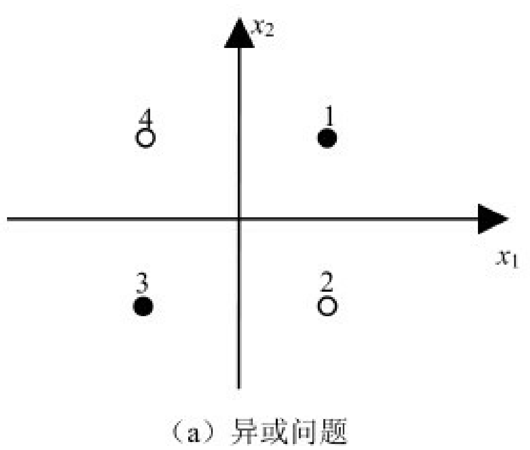
式（15-23）、式（15-24）在形式上同式（15-21）、式（15-17）非常相似，只是将原特征空间中的向量点积替换为原向量在非线性映射空间r^D 中的映像之间的点积。

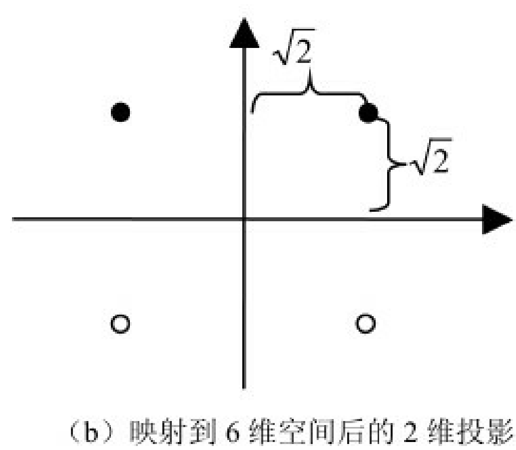
下面我们来看一个通过非线性映射将原本非线性可分问题变为映射空间中线性可分问题的实例。

[例15.1]异或（XOR）问题的SVM

异或问题是最简单的一个无法直接对样本特征向量采用线性判别函数来解决的问题，在$\vec{x}=(1,1)^T$ 的点1和在$\vec{x}=(-1,-1)^T$ 的点3属于类W_1,（图15.9中的实心圆点），在$\vec{x}=(1,-1)^T$ 的点2和在$\vec{x}=(-1, 1)^T$ 的点4属于W_2（空心圆点）。

这样在二维空间r^2中，我们无法找到一条直线（r^2 中的线性分类器）可以将两类样本完全分开。通过使用SVM的方法，利用非线性映射ψ将这4个特征向量映射到更高维的空间r^D 可以解决异或问题，在r^D中这4个特征向量是线性可分的。存在很多这样的ψ函数，这里选用一个最简单的且展开不超过2次的ψ。





▲图15.9 异或问题的非线性映射

$$\vec{x}=(x_1,x_2)\xrightarrow{\psi}\vec{x}^6=(1,\sqrt{2}x_1,\sqrt{2}x_2,\sqrt{2}x_1x_2,{x_1}^2,{x_2}^2),$$

其中，x_i为特征向量$\vec{x}$在第i维上的分量，$\sqrt{2}$是为了规范化。

通过映射ψ，原二维空间中的4个点被分别映射至6维空间。

$$\vec{x}=(1,1)\xrightarrow{\psi}\vec{x}^6=(1,\sqrt{2},\sqrt{2},\sqrt{2},1,1)$$
$$\vec{x}=(1,-1)\xrightarrow{\psi}\vec{x}^6=(1,\sqrt{2},-\sqrt{2},-\sqrt{2},1,1)$$
$$\vec{x}=(-1,-1)\xrightarrow{\psi}\vec{x}^6=(1,-\sqrt{2},-\sqrt{2},\sqrt{2},1,1)$$
$$\vec{x}=(-1,1)\xrightarrow{\psi}\vec{x}^6=(1,-\sqrt{2},\sqrt{2},-\sqrt{2},1,1)$$

图15.9（b）所示是训练样本$\vec{x}$被映射到6维空间后的分布情况，由于无法画出6维空间，图中显示的是样本在第2和第4维度上的二维投影。很明显在这个r^6 空间中可以找到最佳分割超平面$g(x_1,x_2)=x_1x_2=0$，且空白间隔为$\sqrt{2}$，该超平面对应于原始特征空间的双曲线$x_1,x_2=\pm1$。

下面给出求解的过程。

根据式（15-22），约束条件为

$$\sum_{i=1}^{N}\alpha_i y_i=0\Rightarrow\ \alpha_1-\alpha_2+\alpha_3-\alpha_4=0,\quad 0\leqslant\alpha_k,(k=1,2,3,4)$$

由于线性可分这里可不考虑代价系数C。在此约束条件下，最大化式（15-23），即

$$L(\alpha)=\sum_{i=1}^{4}\alpha_i-\frac{1}{2}\sum_{i=1}^{4}\sum_{j=1}^{4}\alpha_i\alpha_j y_i y_j {x_i}^6{x_j}^6$$

显然，由于这个问题的对称性，可取$\alpha_1=\alpha_3$，$\alpha_2=\alpha_4$。而且对于这个简单的问题，我们不必采用标准的二次规划技术，而是可以直接用解析

的方法求解，将训练样本的特征向量$\vec{x}$和对应的类标签y代入上式，解得：α_k^*=1/8，k=1,2,3,4，从而可知这4个训练样本都是支持向量。

注意

大多数SVM解决的问题支持向量总是远远少于样本总数，但由于例15.1中的异或（XOR）问题具有高度的对称性，4个训练样本均为支持向量。

3．核函数（**Kernel**）

例15.1成功地利用了非线性映射ψ解决了在原二维空间中非线性可分的异或问题，然而这只是一个简单的例子，计算所有样本的非线性映射并在高维空间中计算其点积常常是困难的。幸运的是，在一般情况下我们都不必如此，甚至不需要去关心映射 ψ 的具体形式。注意到在上面的对偶问题中, 不论是式（15-23）的寻优目标函数式还是式（15-24）的决策（分类）函数都只涉及样本特征向量之间的点积运算 ψ（x_i）·ψ（x_j）。因此, 在高维空间实际上只需进行点积运算, 而再高维数的向量的点积结果也是一个常数，那么能否抛开映射 ψ（x_i）和 ψ（x_j）的具体形式而直接根据x_i和x_j在原特征空间中得到ψ（x_i）·ψ（x_j）的常数结果呢？答案是肯定的，因为这种点积运算是可以用原特征空间中的核函数（**kernel**）实现的。

根据泛函的有关理论, 只要一种核函数K(x_i, x_j)满足Mercer条件, 它就对应某一变换空间（r^D）中的内积，这似乎很神奇。下面就引入核函数的概念。

核函数是一个对称函数K：$r^n \times r^n \rightarrow$ R，它将两个r^n 空间中的n维向量映射为一个实数。Mercer核函数计算高维空间中的点积。

$$K(x_i, x_j) = \psi(x_i)\cdot\psi(x_j)，\text{其中}\ \psi: R^n \rightarrow R^D \qquad (15\text{-}26)$$

这样如果能够在特征空间中发现计算点积的Mercer 核函数，就可以

使用该核函数代替支持向量机中的点积运算，而根本不用去关心非线性映射ψ的具体形式，因为在SVM训练和分类中的所有相关公式中，ψ都没有单独出现过，总是以ψ（x_i）·ψ（x_j）的形式出现。

因此采用适当的内积核函数K(x_i, x_j)就可以实现从低维空间向高维空间的映射，从而实现某一非线性分类变换后的线性分类,而计算复杂度却没有增加。此时式（15-23）的优化目标函数变为

$$L(\alpha)=\sum_{i=1}^{N}\alpha_i-\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j K(x_i,x_j) \quad （15\text{-}27）$$

而式（15-24）的决策（分类）函数也变为

$$h(x)=\mathrm{sgn}(\sum_{i\in SV}\alpha_i^* y_i K(x_i,x_j)+b^*) \quad （15\text{-}28）$$

常用的核函数如下。

- ● 线性核函数：K(x,y)=x·y （15-29）
- ●多项式核函数：K(x,y)=(x·y+1) ^{d} , d=1, 2, ... （15-30）
- ●径向基核函数：K(x,y)=exp(−γ||x−y|| ^{2}) （15-31）
- ● Sigmoid核函数：K(x, y)=tanh(b(x·y)−c) （15-32）

15.4.4 小节将对这几种常用的核函数的各自特点以及如何选择适用于特定问题的核函数进行介绍，并对最为常用的径向基核函数的使用进行详细的阐述。

15.2.4 推广到多类问题

在之前的问题描述中读者可能也已经注意到，SVM是一个二分器，只能用于2类样本的分类。当然如果仅仅如此，SVM 是不可能得到如此广泛的应用的。下面就来研究如何将二分 SVM 进行推广，使其能够处理多类问题。

有3种常用的策略可用于推广SVM解决多类问题，下面以一个4类问题为例进行说明。

1．一对多的最大响应策略（**one against all**）

假设有A、B、C、D4类样本需要划分。在抽取训练集的时候，分别按照如下4种方式划分。

● A 所对应的样本特征向量作为正集（类标签为+1），B、C、D 所对应的样本特征向量作为负集（类标签为–1）。

● B所对应的样本特征向量作为正集，A、C、D所对应的样本特征向量作为负集。

● C所对应的样本特征向量作为正集，A、B、D所对应的样本特征向量作为负集。

● D所对应的样本特征向量作为正集，A、B、C所对应的样本特征向量作为负集。

对上述4个训练集分别进行训练，得到4个SVM分类器。在测试的时候，把未知类别的测试样本$\vec{x}$分别送入这4个分类器进行判决，最后每个分类器都有1个响应，分别为$f_1(\vec{x})$、$f_2(\vec{x})$、$f_3(\vec{x})$、$f_4(\vec{x})$，最终的决策结果为$\max(f_1(\vec{x}),f_2(\vec{x}),f_3(\vec{x}),f_4(\vec{x}))$，即4个响应中的最大者。

注意这里所说的响应是指决策函数$h(x)=\mathrm{sgn}(\vec{w}\cdot\psi(\vec{x})+b)$在符号化之前的输出$f(\vec{x})=\vec{w}\cdot\psi(\vec{x})+b$，h($\vec{x}$)表示$\vec{x}$位于分割超平面的哪一侧，只反映了$\vec{x}$的类别，而f($\vec{x}$ $\vec{x}$)还能体现出$\vec{x}$与分割超平面的距离远近（绝对值越大越远），因此它能够反映出样本

属于某一类别的置信度。例如，同样位于分割超平面正侧的2个样本，显然更加远离超平面的样本是正例的可信度较大，而紧贴着超平面的样本则很有可能是跨过分割超平面的一个反例。

2．一对一的投票策略（**one against one with voting**）

将A、B、C、D4类样本两类两类地组成训练集，即 (A,B)、(A,C)、

(A,D)、 (B,C)、 (B,D)、(C,D)，得到6个（对于n类问题，为n(n–1)/2个）SVM二分器。在测试的时候，把测试样本$\vec{x}$依次送入这6个二分类器，采取投票形式，最后得到一组结果。投票是以如下方式进行的。

初始化：vote(A)= vote(B)= vote(C)= vote(D)=0。

投票过程：如果使用训练集(A,B)得到的分类器将$\vec{x}$判定为A类,则vote(A)=vote(A)+1，否则vote(B)=vote(B)+1;如果使用(A,C)训练的分类器将$\vec{x}$判定为A类,则vote(A)=vote(A)+1，否则vote(C)=vote(C)+1;……;如果使用(C,D)训练的分类器将$\vec{x}$判定为C类,则vote(C)=vote(C)+1，否则vote(D)=vote(D)+1。

最终判决：Max(vote(A), vote(B), vote(C), vote(D))。如有两个以上的最大值，则一般可简单地取第一个最大值所对应的类别。

3．一对一的淘汰策略（**one against one with eliminating**）

这是在文献[10]中专门针对 SVM 提出的一种多类推广策略，实际上它也适用用所有可以提供分类器置信度信息的二分器。该方法同样基于1对1判别策略解决多类问题，对于我们的4类问题，须训练 6个分类器: (A,B)、 (A,C)、(A,D)、(B,C)、(B,D)、(C,D)。

显然，对于这4 类中的任意一类,如第A类中的某一样本，就可由(A,B)、(A,C)、(A,D)这3个二分器中的任意一个来识别，即判别函数间存在冗余。于是我们将这些二分器根据其置信度从大到小排序，置信度越大表示此二分器分类的结果越可靠，反之则越有可能出现误判。对这6 个分类器按其置信度由大到小排序并分别编号，假设为1# (A,C)，2# (A,B)，3# (A,D)，4# (B,D)，5# (C,D)， 6# (B,C)。

此时，判别过程如下。

（1）设被识别对象为 x，首先由1 #判别函数进行识别。若判别函数 h(x) = +1，则结果为类型A，所有关于类型C的判别函数均被淘汰；若判别函数 h(x) =–1，则结果为类型C，所有关于类型A的判别函数均被

淘汰；若判别函数 h(x) = 0，为“拒绝决策”的情形，则直接选用 2#判别函数进行识别。假设结果为类型C，则所剩判别函数为 4# (B,D) ,5# (C,D)和 6# (B,C)。

（2）被识别对象x再由4#判别函数进行识别。若结果为类型+1，淘汰所有关于D类的判别函数，则所剩判别函数为6#(B,C)。

（3）被识别对象x再由6#判别函数进行识别。若得到结果为类型+1，则可判定最终的分类结果为B。

那么，如何来表示置信度呢？对于SVM而言，分割超平面的分类间隔越大，就说明2类样本越容易分开，表明了问题本身较好的可分性。因此可以用各个SVM二分器的分类间隔大小作为其置信度。

在上述的一对一的淘汰策略中，每经一个判别函数就有某一类别被排除，与该类别有关的判别函数也被淘汰。因此，一般经过c–1次判别就能得到结果。然而，由于判别函数在决策时有可能会遇到“拒绝决策”的情形（h(x) = 0），此时，若直接令h(x)为+1或–1，而把它归于某一类，就可能导致误差。所以不妨利用判别函数之间所存在的冗余进行再决策，以减少这种因“拒绝决策”而导致的误判，一般再经过1或2次决策即可得到最终结果。

以上3种多类问题的推广策略在实际应用中一般都能取得满意的效果，相比之下第2种和第3种在很多情况下能取得更好的效果，在15.4节基于Matlab的人脸识别系统中我们将使用第2种一对一的投票策略解决多类问题。

提示

由两类向多类的推广不仅仅是在SVM中才会遇到的问题。在模式识别和机器学习领域还有很多天然的二分器，如线性感知器、Adaboost等。一般来说，上面讨论的3种推广策略对于这些二分器同样适用。

15.3 SVM的Matlab实现

Matlab从7.0版本开始提供对SVM的支持，其SVM工具箱主要通过svmtrain和svmclassify两个函数封装了SVM训练和分类的相关功能。这两个函数十分简单易用，即使对于SVM的工作原理不是很了解的人也可以轻松掌握。本节将介绍SVM工具箱的用法并给出一个应用实例。

15.3.1 训练——svmtrain

函数svmtrain用来训练一个SVM分类器，常用的调用语法为

SVMStruct = svmtrain(Training, Group)

参数说明

● Training是一个包含训练数据的m行n列的二维矩阵。每行表示1个训练样本（特征向量）， m表示训练样本数目；n表示样本的维数。

● Group是一个代表训练样本类标签的一维向量。其元素值只能为0或1，通常1表示正例， 0表示反例。Group的维数必须和Training的行数相等，以保证训练样本同其类别标号的一一对应。

返回值

SVMStruct是训练所得的代表SVM分类器的结构体，包含有关最佳分割超平面的种种信息，如α、$\vec{w}$ 和b等；此外，该结构体的SupportVector域中还包含了支持向量的详细信息，可以使用SVMStruct.SupportVector获得它们。而这些都是后续分类所需要的，如在基于1对1的淘汰策略的多类决策时为了计算出置信度，需要分类间隔值，可以通过α计算出 $\vec{w}$ 的值，从而得到分割超平面的空白间隔大小 $m=\frac{2}{\|\vec{w}\|}$ ）。

除上述的常用调用形式外，还可通过<'属性名', 属性值>形式的可选参数设置一些训练相关的高级选项，从而实现某些自定义功能，说明如

下。

1．设定核函数

svmtrain函数允许选择非线性映射时核函数的种类或指定自己编写的核函数，方式如下。

SVMStruct = svmtrain(..., 'Kernel_Function', Kernel_FunctionValue);

其中，参数Kernel_FunctionValue的常用合法取值如表15.1所示。

表15.1 参数Kernel_FunctionValue的合法取值

合 法 取 值	含 义
'linear'	线性核函数（默认选项）
'polynomial'	多项式核函数（默认阶数 d=3，d 的意义参见式（15-30））
'rbf '	径向基核函数
Function handle	以符号“@”开头的自己编写的核函数的句柄

例如我们希望采用径向基核函数训练SVM，可以按照如下方式调用。

SVMStruct = svmtrain（Training, Group, 'Kernel_Function', 'rbf '）;

此外，还可以设置与特定核函数相关的参数。如下面的调用表示选用4阶（d= 4）的多项式核函数来实现映射，训练SVM。

SVMStruct = svmtrain(Training, Group, ‘Kernel_Function’, ‘polynomial’, 'Polyorder', 4);

注意设置的核函数参数必须与选用的核函数保持一致。

自定义核函数：

有时我们需要使用自己的核函数计算在映射空间中的点积，此时可以将参数Kernel_FunctionValue设置为自己编写的核函数 kfun的句柄，以 @kfun 的形式给出。

此时的核函数定义为

function K = kfun(X, Y)

其中，X、Y分别为m行和n行的二维矩阵，它们拥有相同的列数l，即自定义核函数需要计算X中的m个l维向量与Y中的n个l维向量两两之间的核函数值，总共m×n个实数值结果，放在m × n的二维矩阵K中返回。K(i, j)就表示向量X（i, ：）与向量Y（j, ：）在高维映射空间中的点积。

如果要向自定义核函数中传递参数，则核函数应定义为

function K = kfun(X, Y, P1, P2)

其中，X、Y含义同上，而P1、P2为核函数的参数，在调用时以@(X,Y) kfun(X,Y,P1,P2) 的形式给出。例如，要自定义一个式（15-32）中的 Sigmoid 核函数K(x, y)=tanh(b(x·y)−c)，首先建立一个kfun_sigmoid.m文件，其中定义核函数kfun_sigmoid。

```
function K = kfun_sigmoid(X, Y, b, c)
% sigmoid核函数，b，c为其参数
K = tanh(b*(X*Y') c);
```

调用时格式为

SVMStruct= svmtrain(Training, Group, 'Kernel_Function', @(X,Y) kfun_sigmoid(X,Y,1,0));

2．训练结果的可视化

当训练数据是二维时可利用'ShowPlot'选项来获得训练结果的可视化解释，调用形式如下。

svmtrain(..., 'ShowPlot', ShowPlotValue);

此时，只需设置ShowPlotValue的值为1(true)即可。

3．设定错误代价**C**

在15.2.2小节讨论非线性可分情况下的C-SVM时，介绍了错误代价系数C对于训练和分类结果的影响，下面将给出设定C值的方法。由式（15-21）可知，引入C对于二次规划问题求解的影响仅仅体现在约束条件当中，因此通过在调用svmtrain时设置一个优化选项‘boxconstraint’即

可，调用形式为

SVMStruct = svmtrain(...,'boxconstraint',C);

其中的C即为错误代价系数，默认取值为Inf，表示错分的代价无限大，分割超平面将倾向于尽可能最小化训练错误。通过适当地设置一个有限的C 值，将得到一个图 15.7(c)中所示的软超平面。

15.3.2 分类——svmclassify

函数svmclassify利用训练得到的SVMStruct结构对一组样本进行分类，常用调用形式为

Group = svmclassify(SVMStruct, Sample);

参数说明

● SVMStruct是训练得到的代表SVM分类器的结构体，由函数svmtrain返回。

● Sample是要进行分类的样本矩阵，每行为1个样本特征向量，总行数等于样本数目，总列数是样本特征的维数，它必须和训练该SVM时使用的样本特征维数相同。

返回值

● Group是一个包含Sample中所有样本分类结果的列向量，其维数与Sample矩阵的行数相同。

当分类数据是二维时可利用'ShowPlot'选项来获得分类结果的可视化解释，调用形式如下。

svmclassify(..., 'Showplot', ShowplotValue)

15.3.3 应用实例

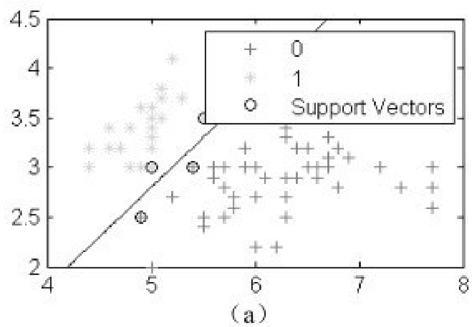
下面的Matlab实例使用svmtrain和svmclassify函数解决了一个二维空间中的2类问题。

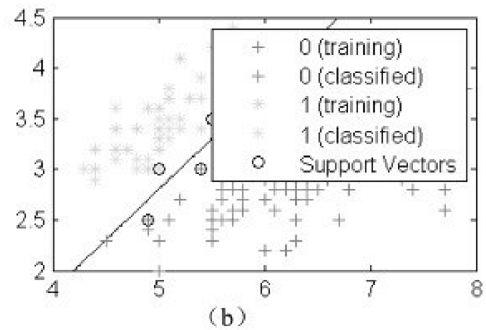
[例15.2] 二维SVM的可视化解释

本例使用Matlab自带的鸢尾属植物数据集来将刚刚学习的SVM训练与分类付诸实践，数据集本身共150个样本，每个样本为一个4维的特征向量，这4维特征的意义分别为花瓣长度、花瓣宽度、萼片长度和萼片宽度。150个样本分别属于3类鸢尾属植物（每类50个样本）。实验中只用了前二维特征，这主要是为了便于训练和分类结果的可视化。为了暂时避开多类问题，我们将样本是哪一类的3类问题变成了样本是不是‘setosa’类的2类问题。

```
load fisheriris % 载入fisheriris数据集
data = [meas(:,1), meas(:,2)]; % 取出所有样本的前二维作为特征
% 转化为“是不是setosa类”的2类问题
groups = ismember(species,'setosa');
% 利用交叉验证随机分割数据集
[train, test] = crossvalind('holdOut',groups);
% 训练一个线性的支持向量机，训练好的分类器保存在svmStruct
svmStruct = svmtrain(data(train,:),groups(train),'showplot',true);
% 利用包含训练所得分类器信息的svmStruct对测试样本进行分类，分类结果保存到classes
classes = svmclassify(svmStruct,data(test,:),'showplot',true);
% 计算测试样本的识别率
ans = nCorrect = sum( classes == groups(test,:) ); % 正确分类的样本数目
accuracy = nCorrect / length(classes) % 计算正确率
accuracy =
0.9867
```

上述程序的运行结果如图15.10所示。





▲图15.10 两类 SVM的训练与分类结果

15.4 综合案例——基于PCA和SVM的人脸识别系统

人脸识别由于其在公安部门、安全验证系统、信用卡验证、档案管理和人机交互系统等方面的广阔应用前景，已经成为当前模式识别和人工智能领域的一个研究热点。本节将向您展现一个极具吸引力的综合案例——基于PCA和SVM的人脸识别系统，在一步步深入问题的同时也给出了使用SVM解决问题的一般框架。

15.4.1 人脸识别简介

人脸识别技术就是以计算机为辅助手段，从静态图像或动态图像中识别人脸。问题一般可以描述为给定一个场景的静态或视频图像，利用已经存储的人脸数据库确认场景中的一个或多个人。一般来说，人脸识别研究一般分为3个部分：从具有复杂背景的场景中检测并分离出人脸所在的区域；抽取人脸识别特征；然后进行匹配和识别。

虽然人类从复杂背景中识别出人脸及表情相当容易，但人脸的自动机器识别却是一个极具挑战性的课题。它跨越了模式识别，图像处理、计算机视觉以及神经生理学、心理学等诸多研究领域。

如同人的指纹一样，人脸也具有唯一性，可用来鉴别一个人的身份，人脸识别技术在商业、法律和其他领域有着广泛的应用。目前，人

脸识别已成为法律部门打击犯罪的有力工具，在毒品跟踪、反恐怖活动等监控中有着很大的应用价值；此外，人脸识别的商业应用价值也正在日益增长，主要是信用卡或者自动取款机的个人身份核对。与利用指纹、手掌、视网膜、虹膜等其他人体生物特征进行个人身份鉴别的方法相比，人脸识别具有直接、友好、方便的特点，特别是对于个人来说没有任何心理障碍。

15.4.2 前期处理

实验数据集仍然采用ORL人脸库[6]。由于每幅人脸图像均包括112×92个像素（参见12.4.1小节），巨大的维数打消了我们将每幅图像直接以其像素作为特征（ANN数字字符识别中的做法）的念头。而实际上，由于原始图像各维像素之间存在着大量的相关性这种做法也是没有必要的。因此需要首先通过主成分分析（PCA）的方法去除相关性，我们将在12.4节基于主成分分析（PCA）的人脸特征提取工作的基础上进行本实验，将PCA降维后得到的20维特征向量作为SVM分类的特征。

前期预处理工作的具体步骤如下。

（1）数据集的分割

将整个数据集分为2个部分 —— 1个训练集和1个测试集。具体地说，我们将每个人的10张面部图像分成2组，前5张放入训练集，另外5张用作测试。这样训练集与测试集各有40×5 = 200个人脸样本。

（2）读入训练图像

将每张图像按列存储为1个 10 304维的行向量。这样 400个人共组成一个 400×10 304的二维矩阵faceContainer，每行1个人脸样本。readFeacs()函数封装了上述功能，由于在12.4.2小节已经介绍过该函数，这里只给出它的原型。

function

[imgRow,imgCol,FaceContainer,faceLabel]=ReadFaces(nFacesPerPerson, nPerson,

bTest)

// 调用时bTest为true（1）表示读入测试样本；默认为false（0），读入训练样本

（3）利用PCA降维去除像素之间的相关性

从全部的训练样本中提取主成份，实验中将主成分的数目确定为20，正如 12.4.3 小节所做的那样。通过投影完成基的转换，每个10 304维的人脸向量被降至20维，在后续的计算中将以此 20维的特征向量来代表该人脸样本。这些工作由我们熟悉的函数 fastPCA()完成，其详细用法和实现细节请参见12.3.4小节。

15.4.3 数据规格化

数据规格化（Scaling）又称数据尺度归一化，就是将特征的某个属性（特征向量的某一维）的取值范围投射到一个特定范围之内，以消除数值型属性因大小范围不一而影响基于距离的分类方法结果的公正性。

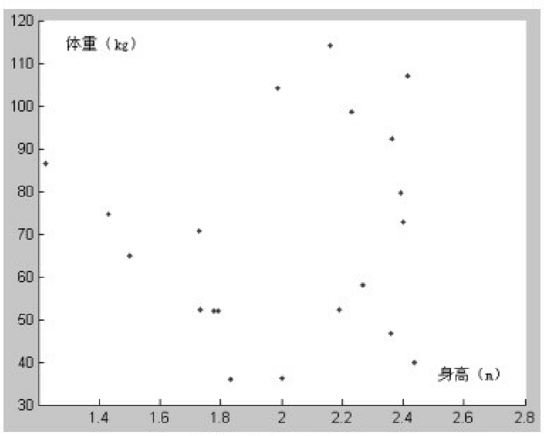
1．数据规格化的必要性

可以毫不夸张地说Scaling在一个模式识别问题中占据着举足轻重的地位，甚至关系到整个识别系统的成败。然而不幸的是，如此重要的一个环节却往往易被初学者忽视。当您在同一个数据集上应用了相同的分类器却得到远不如他人的结果时，首先请确定您是否进行了正确的Scaling。通常进行Scaling一般有以下两点必要性。

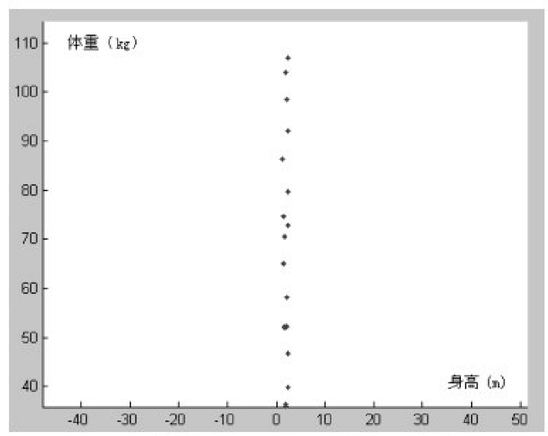
（**1**）防止那些处在相对较大的数字范围（**numeric ranges**）的特征压倒那些处在相对较小的数字范围的特征。举例来说，我们拿到了一份体检报告的数据：其中的一个特征为身高，单位为米（m），这样该特征的数字范围可能就在[1.2, 2.5]，相对于用公斤（kg）作为单位的

体重特征[35, 120]来说，身高特征无疑是处在一个相对很小的数字范围。而很多分类器，包括SVM，都是基于欧氏距离的，这样如果选用这二维特征进行分类，处在小范围的身高由于对距离的计算没有什么贡献，分类器将几乎只根据体重特征来分类而不考虑样本间身高的差异。

图15.11给出了这种情况下的样本二维空间分布，这有助于理解数字范围差异是如何影响距离计算，进而影响基于欧氏距离的分类器的。很明显在图15.11（b）中，样本之间几乎体现不出身高差异，假设两个人A、B分别高1.6m和2m，体重相等；另外两人C、D身高相等，体重分别为75kg和76kg。此时基于欧氏距离的分类器会认为A、B两人更为相似（空间欧氏距离为0.4），而C、D两人则差异较大（空间欧氏距离为1）。这显然与我们常规的判断相反。



（a）身高坐标轴被放大显示



（b）身高与体重以相同的坐标尺度被显示

▲图15.11 20个人的体检数据样本在身高-体重二维空间分布，为便于说明问题样本点是等概率随机生成而不像一般真实数据那样成正态分布

（**2**）避免计算过程中可能会出现的数字问题。例如在计算SVM的核函数时通常需要计算特征向量的内积（线性核函数以及多项式核函数），较大的特征取值可能造成最终的内积结果太大以至于超出计算机的表示范围而溢出。针对这种情况我们建议线性地缩放各个属性到统一

的范围[-1，+1]或[0，1]。

还想顺便提及的一种需要Scaling解决的数字问题是在很多需要计算复合概率情况下（马尔可夫模型-Marcov Model），此时需要计算很多值的连乘积，由于概率取值都在 0～1之间，这有很可能造成连乘积太小以至高精度浮点数都无法表示而下溢出。这就需要在每步运算之后对数据进行等比例的缩放，使计算中间结果始终保持在表示精度的范围以内。

2．数据规格化方法

在训练之前，需要对训练集中的全体样本进行规格化。一般来说，有以下两种常见的数据规格化策略。

（1）最大最小规格化方法

该方法对被初始数据进行一种线性转换。设min_A和max_A分别为属性A的最小和最大值。最大最小规格化方法将属性A的一个值v映射为v’且有v’∈[new_min_A,new_max_A]。具体映射计算公式如下。

$$v' = \frac{v - \min_A}{\max_A - \min_A}(new_\max_A - new_\min_A) + new_\min_A \qquad (15\text{-}33)$$

例如，属性“体重”的最大最小值分别是40kg和120kg，利用最大最小规格化方法将属性A的值映射到[-1,1]的范围内，那么对属性A的值75kg将被转化为

$$v' = \frac{75-40}{120-40}(1-(-1)) + (-1) = -0.125$$

（2）零均值规格化方法

该方法是根据属性A的均值和偏差来对A进行规格化，可将训练集中的每个样本特征的均值统一变换为0，并且都具有统一的方差（如1.0）。属性A的v值可以通过以下计算公式获得其映射值v’。

$$v' = \frac{v - \mu_A}{\sigma_A} \qquad (15\text{-}34)$$

式中的μ_A和σ_A分别为属性A的均值和方差。

提示

零均值规格化方法常用于属性A的最大值与最小值未知的情况。

当然，在测试阶段我们必须对测试样本应用同样的Scaling技术。比如说采用最大最小规格化方法将训练数据某一维从[-10，+10]线性缩放至[-1，+1]，那么也要对测试数据的该维应用相同的变换规则$-1+\frac{x-(-10)}{10-(-10)}*(1-(-1))$，即如测试数据在该维上处在范围[-11，8]，应缩放至[−1.1，+0.8]。

3．实现人脸特征数据的规格化

正是基于前述原因，在将降维后的数据交给SVM处理之前，首先需要进行Scaling。这里选择第一种方法，线性地缩放特征的各个属性（维度）到 [–1，+1]。

人脸特征数据规格化的完整实现如下，它被封装在金羽图书论坛（http://bbs.book95.com）的“金羽图书与答疑”板块与本书同名的主题帖子附件的“chapter15\code\FaceRec”目录下的 scaling.m文件中。

function [SVFM, lowVec, upVec] = scaling(VecFeaMat, bTest, lRealBVec, uRealBVec)

% Input: VecFeaMat --- 需要scaling的 m*n 维数据矩阵，每行一个样本特征向量，列数为维数

% 　　bTest --- =1：说明是对于测试样本进行scaling，此时必须提供 lRealBVec 和 uRealBVec

% 　　　　的值，此二值应该是在对训练样本 scaling 时得到的

% 　　　　=0：默认值，对训练样本进行 scaling

```
%        lRealBVec --- n维向量，对训练样本 scaling 时得到的各维的实际下限信息lowVec
%        uRealBVec --- n维向量，对训练样本 scaling 时得到的各维的实际上限信息upVec
%
% output: SVFM --- VecFeaMat的 scaling 版本
%          upVec --- 各维特征的上限(只在对训练样本scaling时有意义，bTest = 0)
%          lowVec --- 各维特征的下限(只在对训练样本scaling时有意义，bTest = 0)
if nargin < 2
  bTest = 0;
end
% 缩放目标范围[-1, 1]
lTargB = -1;
uTargB = 1;
[m n] = size(VecFeaMat);
SVFM = zeros(m, n);
if bTest
  if nargin < 4
    error('To do scaling on testset, param lRealB and uRealB are needed.');
end
if nargout > 1
  error('When do scaling on testset, only one output is supported.');
end
for iCol = 1:n
```

```
        if uRealBVec(iCol) == lRealBVec(iCol)
          SVFM(:, iCol) = uRealBVec(iCol);
          SVFM(:, iCol) = 0;
        else
          SVFM(:, iCol) = lTargB + ( VecFeaMat(:, iCol) - lRealBVec(iCol) )
/
      ( uRealBVec(iCol)-lRealBVec(iCol) ) * (uTargB-lTargB); % 测试数据
的scaling
            end
          end
        else
       upVec = zeros(1, n);
      lowVec = zeros(1, n);
       for iCol = 1:n
         lowVec(iCol) = min( VecFeaMat(:, iCol) );
         upVec(iCol) = max( VecFeaMat(:, iCol) );
         if upVec(iCol) == lowVec(iCol)
            SVFM(:, iCol) = upVec(iCol);
            SVFM(:, iCol) = 0;
         else
            SVFM(:, iCol) = lTargB + ( VecFeaMat(:, iCol) - lowVec(iCol) ) /
      ( upVec(iCol)-lowVec(iCol) ) * (uTargB-lTargB); % 训练数据的
scaling
             end
         end
      end
```

15.4.4 核函数的选择

到目前为止，要送入 SVM 的数据已经准备就绪，但在“启动”SVM 让它为我们工作之前仍有两个问题摆在面前：① 选择哪一种核函数（Kernel）；② 确定核函数的参数以及错误代价系数C的最佳取值。下面先来解决第1个问题，第2个问题留到15.4.5小节中。

由于只有4种常用的核函数（参见15.2.3小节），将它们依次尝试并选择对测试数据效果最好的一个似乎也是个行得通的方法，但后续的参数选择问题将使这成为一个复杂的排列组合问题，而远远不是4种可能那么简单。

尽管最佳核函数的选择一般与问题自身有关，但还是有规律可循的。建议初学者在通常情况下优先考虑径向基核函数（RBF）。

$$K(x,y)=\exp(-\gamma\|x-y\|^2)$$

这主要基于以下考虑。

● 作为一种对应于非线性映射的核函数，RBF能够处理非线性可分的情况。

● 线性核函数是RBF核函数的一种特殊情况，即通过适当地选择参数（γ,C），RBF核函数总可以得到与带有错误代价参数C的线性核函数相同的效果，反之当然不成立。

● 在选择某些参数的情况下，Sigmoid 核函数$K(x, y)=\tanh(b(x\cdot y)-c)$的行为也类似于 RBF核函数，而且选择Sigmoid核函数就有2个与之有关的参数b、c需要确定。

● 多项式核函数需要计算内积，而这有可能产生溢出之类的计算问题。

工具箱中的RBF核函数无法灵活地选择参数，为了方便设置参数γ，我们编写了自己的RBF核函数kfun_rbf()，函数的完整实现位于金羽

图书论坛（http://bbs.book95.com）的"金羽图书与答疑"板块与本书同名的主题帖子附件的"chapter15\code\FaceRec\Kernel"目录下的kfun_rbf.m文件中。

```
function K = kfun_rbf(U, V, gamma)
% rbf 核函数
[m1 n1] = size(U);
[m2 n2] = size(V);
K = zeros(m1, m2);
for ii = 1:m1
  for jj = 1:m2
    K(ii, jj) = exp( -gamma * norm(U(ii, :)-V(jj, :))^2 );
  end
end
```

15.4.5 参数选择

在选择了RBF核的情况下总共有2个参数需要确定，RBF核自身的参数γ以及错误代价系数C。这个问题本身就是一个优化问题，变量是C和γ，目标函数值就是SVM对于测试集的识别率。困难在于很难用变量C 和 γ 写出目标函数的表达式，因此不适于采用一般的优化策略。幸好在LibSVM（参见 15.4.2小节）中林智仁(Chih-Jen Lin)博士为我们提供了一个非常实用的基于交叉验证和网格搜索的参数选择方法，并提供了相应的工具grid.py，具体可参见文献[13]。随后将结合人脸识别的问题给出参数搜索工具的使用方法，您将学习到如何在基于Matlab的SVM应用中获得工具grid.py的有力帮助。

1．数据集格式化

要利用LibSVM的参数选择工具grid.py，首先需要把数据集格式化

为grid.py所要求的形式，即为如下格式的文本文件。

<类标签> <特征索引 1>:<特征值 1> <特征索引 2>:<特征值 2> ... <特征索引 n>:<特征值 n>

...

...

...

其中，每行对应一个样本实例。对于分类问题，<类标签>是一个表示类别标号的整数，其后的“<特征索引>:<特征值>”对给出了每一维的特征取值，<特征索引>是一个从1开始逐渐递增的整数，<特征值>是一个实数。

我们编写的export()函数用于从Matlab导出LibSVM能够使用的数据，它被封装在金羽图书论坛（ http://bbs.book95.com ）的“金羽图书与答疑”板块与本书同名的主题帖子附件的“chapter15\code\FaceRec\exportLibSVM”目录下的export.m文件中。

```
function export(strMat, strLibSVM)
% 将以参数 strMat 指定的文件（.mat文件）中的数据导出为能够被LibSVM使用的格式（.txt文件），生成的
文件名由参数 strLibSVM 指定
%
% 输入：strMat --- 源文件名（包括路径），'.mat'文件，默认为'../Mat/trainData.mat'，其中必须包含
训练数据
%                 TrainData 和类标签 trainLabel，该文件可在训练 SVM 过程中生成
%           strLibSVM --- 目标文件名（包括路径），'.txt'文件，默认为'trainData.txt'
if nargin < 1
```

```
    strMat = '../Mat/trainData.mat';
    strLibSVM = 'trainData.txt';
elseif nargin < 2
    strLibSVM = 'trainData.txt';
end
[fid, fMsg ] = fopen(strLibSVM, 'w'); % 建立目标输出文件
if fid == -1
    disp(fMsg );
    return
end
strNewLine = [13 10]; % 换行
strBlank = ' ';
load(strMat)
[nSamp, nDim] = size( TrainData );
for iSamp = 1:nSamp
    fwrite(fid, num2str(trainLabel(iSamp)), 'char');
    for iDim = 1:nDim
        fwrite(fid, strBlank, 'char');
        fwrite(fid, [num2str(iDim) ':'], 'char');
        fwrite(fid, num2str(TrainData(iSamp, iDim)), 'char');
    end
    fwrite(fid, strNewLine, 'char');
end
fclose(fid);
```

上述程序执行后，第2、第3个人的人脸样本对应于trainData.txt中的如下2行。

......

2 1:0.17762 2:0.054549 3:0.68467 4:-0.032753 5:-0.076314 6:-0.1758 7:0.37941 8:-0.039629

9:-0.54754 10:0.091465 11:0.13723 12:0.19647 13:0.36674 14:-0.50934 15:0.7817 16:-0.22577

17:0.070353 18:0.10225 19:-0.26392 20:0.01075

3 1:-0.10874 2:-0.239 3:-0.16797 4:-0.26408 5:0.53417 6:0.48635 7:-0.11967 8:-0.051977

9:-0.030884 10:-0.60206 11:-0.27395 12:-0.36117 13:0.23001 14:0.082984 15:0.13227

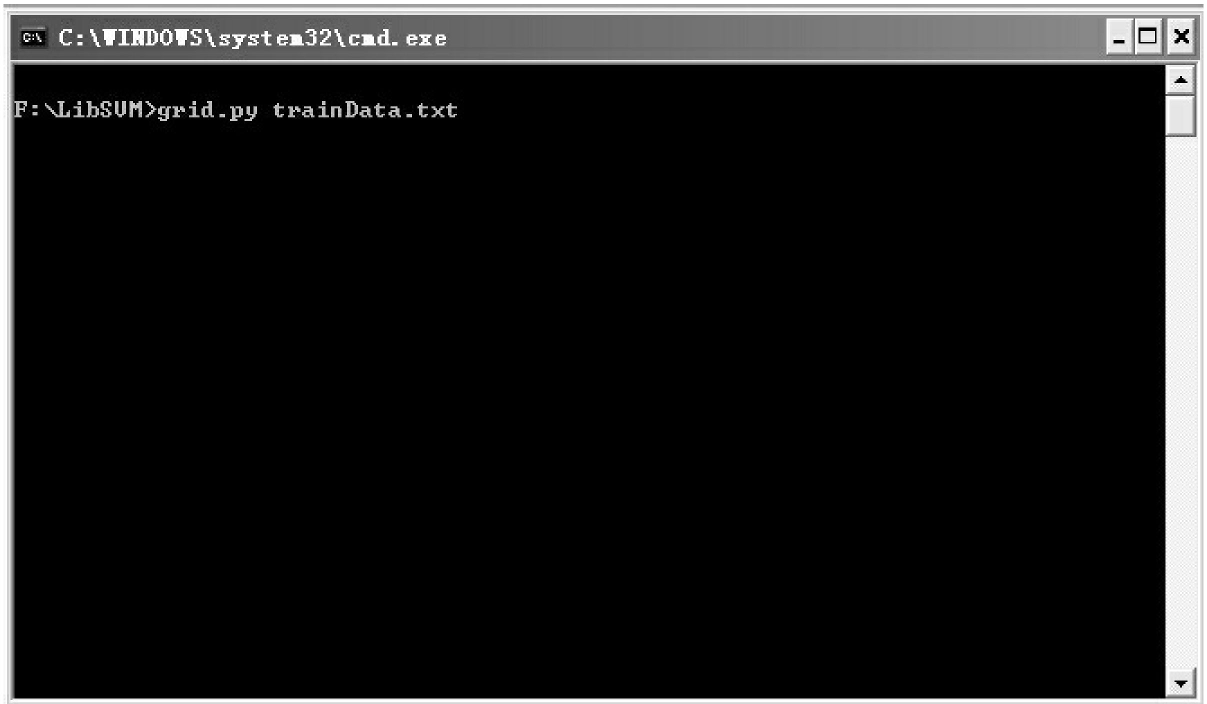
16:0.090856 17:-0.25932 18:0.094344 19:-0.59063 20:0.90311

......

2．搜索参数

参数搜索工具grid.py是python的脚本文件，所以首先系统必须安装python。此外搜索过程中还要用到工具gnuplot.exe以便将搜索过程可视化。关于python和gnuplot您都可以在互联网上找到。

在开始菜单运行命令窗口中输入cmd命令，打开命令提示符后，转移至grid.py所在的目录，在命令行中输入命令“grid.py trainData.txt”，如图 15.12所示。



▲图15.12 运行 grid.py

如果命令运行中出现问题，很可能是 grid.py 中的路径设置需要修正，以便 grid.py 可以找到svmtrain.exe和pgnuplot.exe的位置。图15.13给出了路径设置的方法：打开grid.py文件，修改字母“r”后面引号中的路径到svmtrain.exe和pgnuplot.exe所在的路径即可。图中的路径设置对应于svmtrain.exe和pgnuplot.exe与grid.py处在相同目录下的情况。

```
#!/usr/bin/env python

import os, sys, traceback
import Queue
import getpass
from threading import Thread
from string import find, split, join
from subprocess import *

# svmtrain and gnuplot executable

is_win32 = (sys.platform == 'win32')
if not is_win32:
       svmtrain_exe = "../svm-train"
       gnuplot_exe = "/usr/bin/gnuplot"
else:
       # example for windows
       svmtrain_exe = r"svmtrain.exe"
       gnuplot_exe = r"pgnuplot.exe"

# global parameters and their default values
```

▲图15.13 grid.py中的路径设置

回车后开始自动参数搜索，搜索结果如图15.14倒数第2行所示，得到最佳的C值为128，最佳γ（图中为g）为 0.007 812 5，训练集上交叉验证的识别率为 97.5%。

```
C:\WINDOWS\system32\cmd.exe
[local] -1 -3 91.0  (best c=128.0, g=0.0078125, rate=97.5)
[local] 11 -3 97.0  (best c=128.0, g=0.0078125, rate=97.5)
[local] -3 -3 91.0  (best c=128.0, g=0.0078125, rate=97.5)
[local] 9 -3 97.0  (best c=128.0, g=0.0078125, rate=97.5)
[local] 3 -3 97.0  (best c=128.0, g=0.0078125, rate=97.5)
[local] 15 -3 97.0  (best c=128.0, g=0.0078125, rate=97.5)
[local] -5 -3 91.0  (best c=128.0, g=0.0078125, rate=97.5)
[local] 7 -3 97.0  (best c=128.0, g=0.0078125, rate=97.5)
[local] 1 -3 96.0  (best c=128.0, g=0.0078125, rate=97.5)
[local] 13 -7 97.5  (best c=128.0, g=0.0078125, rate=97.5)
[local] 13 -1 96.5  (best c=128.0, g=0.0078125, rate=97.5)
[local] 13 -13 97.5  (best c=128.0, g=0.0078125, rate=97.5)
[local] 13 1 81.5  (best c=128.0, g=0.0078125, rate=97.5)
[local] 13 -11 97.5  (best c=128.0, g=0.0078125, rate=97.5)
[local] 13 -5 97.5  (best c=128.0, g=0.0078125, rate=97.5)
[local] 13 -15 95.0  (best c=128.0, g=0.0078125, rate=97.5)
[local] 13 3 57.0  (best c=128.0, g=0.0078125, rate=97.5)
[local] 13 -9 97.5  (best c=128.0, g=0.0078125, rate=97.5)
[local] 13 -3 97.0  (best c=128.0, g=0.0078125, rate=97.5)
128.0 0.0078125 97.5

F:\LibSVM>_
```

▲图15.14 grid.py中的路径设置

15.4.6 构建多类SVM分类器

15.2.4小节已经介绍了如何应用1对1的投票策略将SVM推广至多类问题，下面给出该方法的实现细节。我们编写了函数multiSVMTrain()和multiSVMClassify()作为标准SVM工具箱的扩展，从而得到可以解决多类问题的SVM。

1．多类问题的训练

在多类SVM训练阶段，我们要做的就是用 n=40类样本构建n(n - 1)/2个SVM二分器，把每个SVM二分器的训练结果（SVMStruct结构体）都保存到一个结构体的细胞数组CASVMStruct中，具体地说CASVMStruct{ii}{jj}中保存着第ii类与第jj类两类训练得到的SVMStruct。最终将多类SVM分类时需要的全部信息保存至结构体multiSVMStruct中返回，可以说multiSVMStruct中包含了我们的训练成果。

multiSVMTrain()的完整实现如下，它被封装在金羽图书论坛（http://bbs.book95.com）的“金羽图书与答疑”板块与本书同名的主题帖子附件的“chapter15\code\FaceRec\SVM”目录下的multiSVMTrain.m文件中。

```
function multiSVMStruct = multiSVMTrain(TrainData, nSampPerClass, nClass, C, gamma)
% 采用1对1投票策略将 SVM 推广至多类问题的训练过程，将多类SVM训练结果保存至multiSVMStruct中
%
% 输入:--TrainData:每行是一个样本人脸
%       --nClass:人数，即类别数
```

%　　--nSampPerClass:nClass*1维的向量，记录每类的样本数目，如 nSampPerClass(iClass)

%　　 给出了第iClass类的样本数目

%　　--C:错误代价系数，默认为 Inf

%　　--gamma:径向基核函数的参数 gamma，默认值为1

%

% 输出:--multiSVMStruct:一个包含多类SVM训练结果的结构体

% 默认参数

if nargin < 4

  C = Inf;

  gamma = 1;

elseif nargin < 5

  gamma = 1;

end

%开始训练，需要计算每两类间的分类超平面，共(nClass-1)*nClass/2个

for ii=1:(nClass-1)

  for jj=(ii+1):nClass

    clear X;

    clear Y;

    startPosII = sum(nSampPerClass(1:ii-1)) + 1;

    endPosII = startPosII + nSampPerClass(ii) - 1;

    X(1:nSampPerClass(ii), :) = TrainData(startPosII:endPosII, :);

    startPosJJ = sum(nSampPerClass(1:jj-1)) + 1;

    endPosJJ = startPosJJ + nSampPerClass(jj) - 1;

    X(nSampPerClass(ii)+1:nSampPerClass(ii)+nSampPerClass(jj), :) = TrainData(start

```
PosJJ:endPosJJ, :);
      % 设定两两分类时的类标签
      Y = ones(nSampPerClass(ii) + nSampPerClass(jj), 1);
      Y(nSampPerClass(ii)+1:nSampPerClass(ii)+nSampPerClass(jj)) = 0;
      % 第ii个人和第jj个人两两分类时的分类器结构信息
      CASVMStruct{ii}{jj}= svmtrain( X, Y, 'Kernel_Function', @(X,Y)
kfun_rbf(X,Y,gamma),
'boxconstraint', C );
   end
end
% 已学得的分类结果
multiSVMStruct.nClass = nClass;
multiSVMStruct.CASVMStruct = CASVMStruct;
% 保存参数
save('Mat/params.mat', 'C', 'gamma');
```

2．多类问题的分类

在多类SVM分类阶段，让测试样本依次经过训练得到的 n(n -1)/2个（n = 40）个SVM二分器，通过投票决定其最终类别归属。

multiSVMClassify()的实现如下，它被封装在金羽图书论坛（http://bbs.book95.com）的“金羽图书与答疑”板块与本书同名的主题帖子附件的“chapter15\code\FaceRec\SVM”目录下的multiSVMClassify.m文件中。

```
function class = multiSVMClassify(TestFace, multiSVMStruct)
% 采用1对1投票策略将 SVM 推广至多类问题的分类过程
% 输入:--TestFace:测试样本集。m*n 的2维矩阵，每行一个测试样本
%      --multiSVMStruct:多类 SVM 的训练结果，由函数
```

multiSVMTrain 返回，默认是从

Mat/multiSVMTrain.mat文件中读取

%

% 输出:--class: m*1 列向量，对应 TestFace 的类标签

% 读入训练结果

if nargin < 2

t = dir('Mat/multiSVMTrain.mat');

if length(t) == 0

error('没有找到训练结果文件，请在分类以前首先进行训练！');

end

load('Mat/multiSVMTrain.mat');

end

nClass = multiSVMStruct.nClass; % 读入类别数

CASVMStruct = multiSVMStruct.CASVMStruct; % 读入两两类之间的分类器信息

%%%%%%%%%%%%投票策略解决多类问题%%%%%%%%%%%%%%

m = size(TestFace, 1);

Voting = zeros(m, nClass); % m个测试样本，每个样本nPerson 个类别的投票箱

for iIndex = 1:nClass-1

for jIndex = iIndex+1:nClass

classes = svmclassify(CASVMStruct{iIndex}{jIndex}, TestFace);

% 投票

Voting(:, iIndex) = Voting(:, iIndex) + (classes == 1);

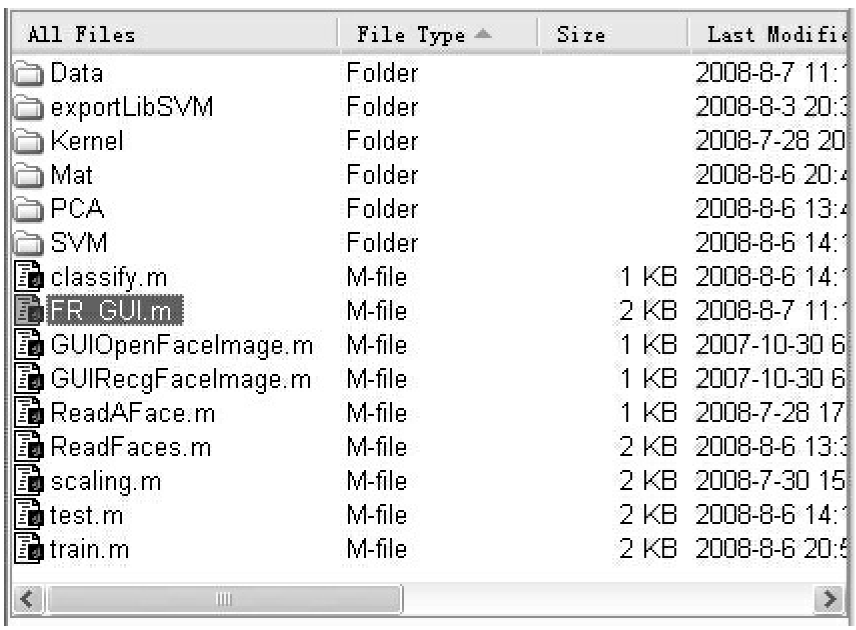
Voting(:, jIndex) = Voting(:, jIndex) + (classes == 0);

end % for jClass

```
end % for iClass
% final decision by voting result
[vecMaxVal, class] = max( Voting , [ ], 2 );
%display(sprintf('TestFace对应的类别是:%d',class));
```

15.4.7 实验结果

终于一切就绪了！相信您已经迫不及待地想看到分类器对于每个人的后5张图片的识别效果了。我们将有关这个人脸识别系统实现的所有文件都存放在金羽图书论坛（http://bbs.book95.com）的“金羽图书与答疑”板块与本书同名的主题帖子附件的“chapter15\code\FaceRec”目录中，作为一个完整的工程。图15.15给出了工程的目录清单。



▲图15.15 FaceRec工程一览

根目录下主要是一些驱动文件，它们的作用是调用封装好的功能函数以完成系统某一部分的特定功能。有些函数像ReadFaces.m以及scaling.m之前已经介绍过了，其余的将在剩下的讨论中给予说明。各个

目录的内容如下。

● Data/：存放着ORL人脸库图像文件（.pgm）。

● exportLibSVM/：包含文件export.m，并存放导出的LibSVM格式的文件。

● Kernel/：存放我们自定义的核函数，现有两个文件kfun_rbf.m和kfun_sigmoid.m。

● Mat/：存放所有的 .mat数据文件。

● PCA/：fastPCA.m和visualize_pc.m，这是我们在第12章中建立的PCA工具箱。

● SVM/：多类SVM工具箱，包括multiSVMTrain.m和multiSVMClassify.m两个文件。

我们编写了图形界面程序FR_GUI以方便您驱动整个实验，实现如下。

```
% FR_GUI.m
global h_axes1;
global h_axes2;
h_f = figure('name', '基于PCA和SVM的人脸识别系统');
h_textC = uicontrol(h_f, 'style', 'text', 'unit', 'normalized', 'string', 'C=',
'position',...
  [0.05 0.7 0.1 0.06]);
h_editC = uicontrol(h_f, 'style', 'edit', 'unit', 'normalized', 'position', [0.05
0.6 0.1
0.06],...
  'callback', 'C = str2num(get(h_editC, ''string''))');
h_textGamma = uicontrol(h_f, 'style', 'text', 'unit', 'normalized', 'string',
'gamma=',
'position',...
```

```
    [0.05 0.5 0.1 0.06]);
  h_editGamma = uicontrol(h_f, 'style', 'edit', 'unit', 'normalized',
'position', [0.05 0.4
  0.1 0.06],...
   'callback', 'gamma = str2num(get(h_editGamma, ''string''))');
  % 取得参数 C 和 gamma 的当前值，即最近一次训练所使用的值
  t = dir('Mat/params.mat');
  if length(t) == 0
    % 没有找到参数文件
    C = Inf;
    gamma = 1
  else
    load Mat/params.mat;
  end
  set(h_editC, 'string', num2str(C));
  set(h_editGamma, 'string', num2str(gamma));
  h_axes1 = axes('parent', h_f, 'position', [0.25 0.23 0.32 0.6], 'visible',
'off');
  h_axes2 = axes('parent', h_f, 'position', [0.62 0.23 0.32 0.6], 'visible',
'off');
  h_btnOpen = uicontrol(h_f, 'style', 'push', 'string', '打开', 'unit',
'normalized',...
  'position', [0.32 0.1 0.18 0.1], 'callback', 'GUIOpenFaceImage');
  h_btnRecg = uicontrol(h_f, 'style', 'push', 'string', '识别', 'unit',
'normalized',...
    'position', [0.67 0.1 0.18 0.1], 'callback', 'GUIRecgFaceImage');
  h_btnRecg = uicontrol(h_f, 'style', 'push', 'string', '训练', 'unit',
```

'normalized',...
    'position', [0.32 0.83 0.18 0.1], 'callback', 'train(C, gamma)');
  h_btnRecg = uicontrol(h_f, 'style', 'push', 'string', '测试', 'unit', 'normalized',...
    'position', [0.67 0.83 0.18 0.1], 'callback', 'test');

在Matlab命令窗口中输入以下命令。

```
%将工程所在文件夹FaceRec添加到系统路径列表
>> addpath(genpath('F:\doctor research\Matlab Work\FaceRec'))
>> FR_GUI
```

上述命令运行后将启动了如图15.16所示GUI识别程序。



▲图15.16 识别程序主界面

1．训练

在程序主界面左侧的编辑框可以设置参数C和γ（图中的gamma），由于在开始时还没有进行训练，C和gamma分别取其默认值Inf和1，以后编辑框中总是显示最近一次训练时所采用的C和gamma值。

单击“训练”按钮会调用函数 train(C, gamma)，其中包括了读入人脸数据、PCA降维、数据规格化以及训练多类SVM等功能。

train()函数的完整实现如下，它位于工程目录下的train.m文件中。

```
function train(C, gamma)
% 整个训练过程，包括读入图像，PCA降维以及多类 SVM 训练，各个阶段的处理结果分别保存至文件：
%  将 PCA 变换矩阵 W 保存至 Mat/PCA.mat
%  将 scaling 的各维上、下界信息保存至 Mat/scaling.mat
%  将 PCA 降维并且 scaling 后的数据保存至 Mat/trainData.mat
%  将多类 SVM 的训练信息保存至 Mat/multiSVMTrain.mat
global imgRow;
global imgCol;
display(' ');
display(' ');
display('训练开始...');
nPerson=40;
nFacesPerPerson = 5;
display('读入人脸数据...');
[imgRow,imgCol,FaceContainer,faceLabel]=ReadFaces(nFacesPerPerso
save('Mat/FaceMat.mat', 'FaceContainer')
display('..............................');
```

```
nFaces=size(FaceContainer,1);%样本（人脸）数目
display('PCA降维...');
[pcaFaces, W] = fastPCA(FaceContainer, 20); % 主成分分析PCA
% pcaFaces是200*20的矩阵, 每一行代表一张主成分脸(共40人，每人5张)，每个脸20个维特征
% W是分离变换矩阵, 10304*20 的矩阵
visualize_pc(W);%显示主成分脸
display('..............................');
X = pcaFaces;
display('Scaling...');
[X,A0,B0] = scaling(X);
save('Mat/scaling.mat', 'A0', 'B0');
% 保存 scaling 后的训练数据至 trainData.mat
TrainData = X;
trainLabel = faceLabel;
save('Mat/trainData.mat', 'TrainData', 'trainLabel');
display('..............................');
for iPerson = 1:nPerson
  nSplPerClass(iPerson) = sum( (trainLabel == iPerson) );
end
multiSVMStruct = multiSVMTrain(TrainData, nSplPerClass, nPerson, C, gamma);
display('正在保存训练结果...');
save('Mat/multiSVMTrain.mat', 'multiSVMStruct');
display('..............................');
display('训练结束。');
```

训练结束后，训练结果和参数等相关信息都会自动记入“Mat”文件

夹下的相关 .mat数据文件，以备测试和分类使用。

2．识别

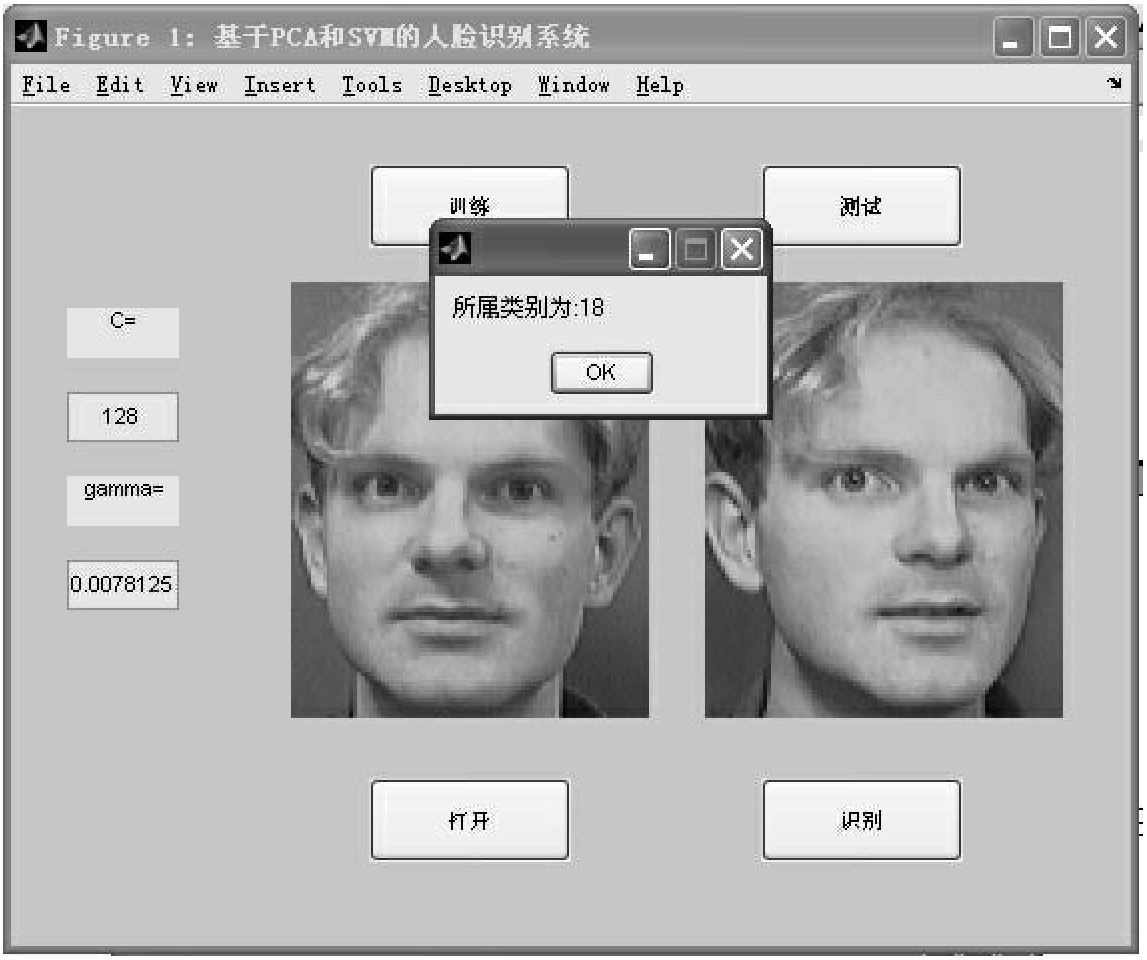
识别是指对某人的身份进行识别，即分类一个特定样本。

首先在图15.16所示的程序主界面中单击“打开”按钮，触发GUIOpenFaceImage过程，在弹出的“打开文件对话框”中选定待识别者的图像后（这里是第18个人的第8幅图像），打开的图像如图15.17所示。

单击“识别”按钮，此时会调用过程GUIRecgFaceImage，进而调用函数classify()。识别结果如图15.18所示，右侧显示的为识别出的人的第1幅图像，可见系统正确识别出此人的身份。



▲图15.17打开待识别者面部图像



▲图15.18通过面部信息进行身份识别

识别中涉及的过程和函数如下。

（1）GUIOpenFaceImage过程

% GUIOpenFaceImage.m

global filepath;

[filename, pathname] = uigetfile({'*.pgm;*.jpg;*tif', ' (*.pgm), (*.jpg), (*.tif)'; ...

'*.*', 'All Files(*.*)' }, 'Select a face image to be recognized');

if filename ～ =0

```
  filepath = [pathname,filename];
  axes(h_axes1);
  imshow(imread(filepath));
end
```

（2）GUIOpenRecgImage过程

```
% GUIRecgFaceImage.m
nClass = classify(filepath);
msgbox( ['所属类别为:',num2str(nClass)] );
axes(h_axes2);
f = imread(['Data/ORL/S',num2str(nClass),'/1.pgm']); % 打开该人的第1幅图像
imshow(f);
```

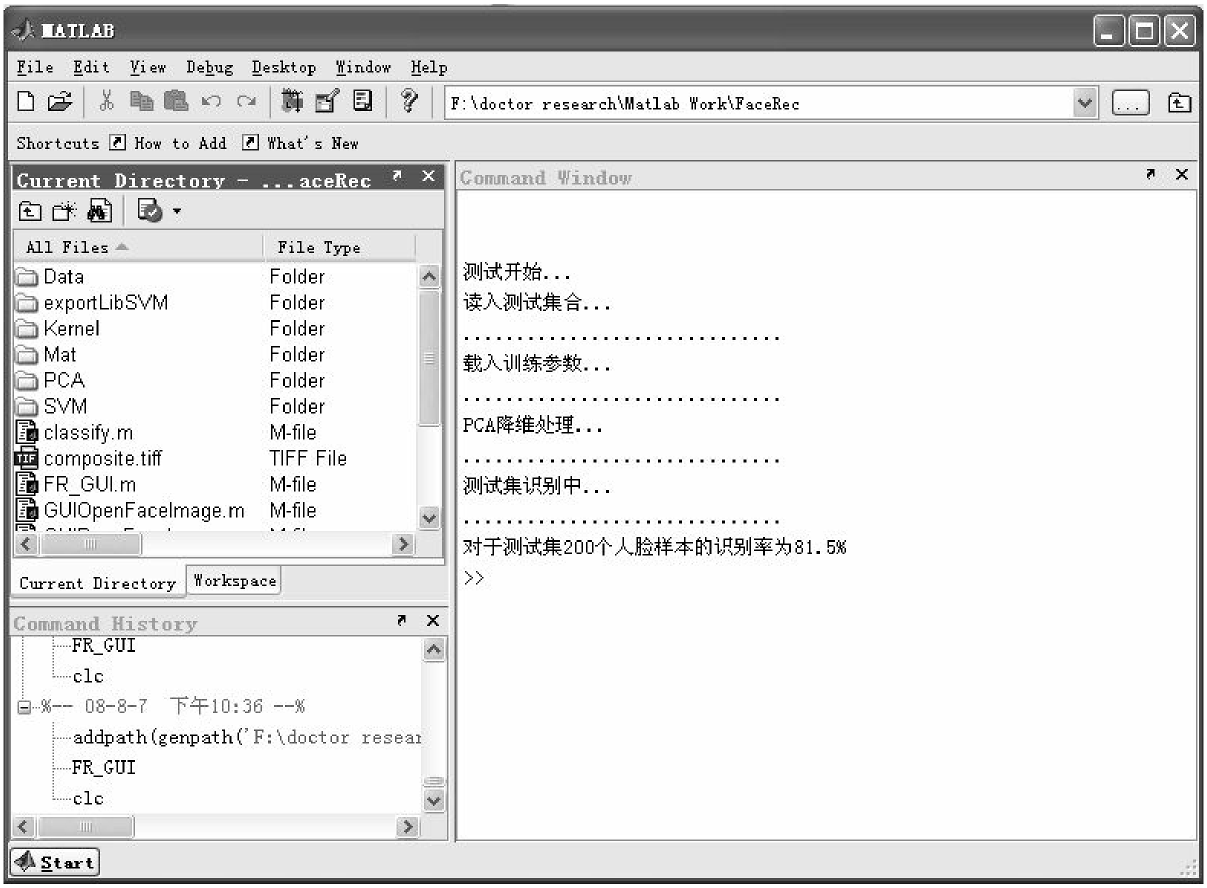
（3）classify( )函数

```
% classify.m
function nClass = classify(newFacePath)
% 整个分类（识别）过程
% 输入：--newFacePath：待识别图像的存取路径
% 输出：--nClass：识别出的类别标号
display(' ');
display(' ');
display('识别开始...');
% 读入相关训练结果
display('载入训练参数...');
load('Mat/PCA.mat');
load('Mat/scaling.mat');
load('Mat/trainData.mat');
load('Mat/multiSVMTrain.mat');
```

```
display('..............................');
xNewFace = ReadAFace(newFacePath); % 读入一个测试样本
xNewFace = double(xNewFace);
xNewFace = xNewFace*W; % 经过pca变换降维
xNewFace = scaling(xNewFace,1,A0,B0);
display('身份识别中...');
nClass = multiSVMClassify(xNewFace);
display('..............................');
display(['身份识别结束，类别为：' num2str(nClass), '。']);
```

3．测试

测试是指分类所有的测试样本（40个人的后5张图像，共200个样本），并计算识别率。在图15.16所示的程序主界面中单击“测试”按钮，将触发test()函数，运行情况如图15.19所示。



▲图15.19 测试结果

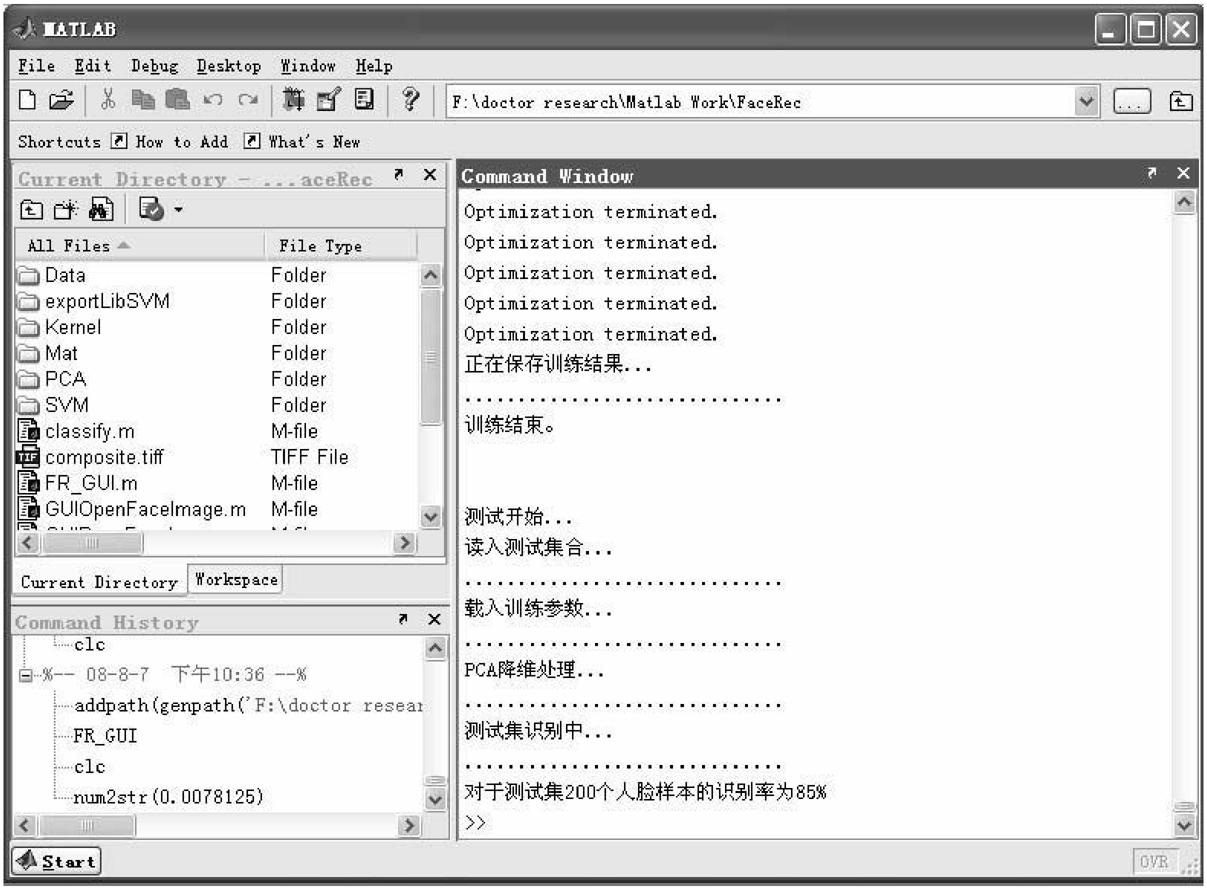
结果显示对于测试集合的200个全新样本，SVM取得了81.5%的识别正确率。考虑到数据集中的前、后5张图片之间存在一定的姿态、表情等因素的差异，这样的识别率是完全可以接受的。要进一步提高识别率，可以从两个方面着手：一是选取更具区分能力（most discriminative）的特征；二是改善分类器本身的性能。

由于在上述训练过程中使用了默认的参数C和gamma，首先想到采用15.4.3小节中得到的最优参数重新训练，以改进我们的多类SVM分类器。设置C为128，gamma为0.007 812 5，如图15.20所示。



▲图15.20 重新训练（C = 128，gamma = 0.007 812 5）

单击“训练”按钮，重新训练分类器，结果如图15.21所示。



▲图15.21 采用优化参数训练后的测试结果（85%）（C = 128，gamma = 0.007 812 5）

我们看到，此时的识别率为85%，高于此前的81.5%，这样就从实验角度证明了参数优化对提高分类器推广能力的作用。

通过选取更具区分能力的特征来提高识别率是一个更为复杂的话题，第12章中我们曾多次强调过所选择的特征对于分类的决定作用。就本节的人脸识别系统而言，可以在PCA处理时尝试不同的维数，观察特征维数对于识别率的影响，当然也会存在一个所谓的最佳维数。此外，在 12.5节中介绍LBP特征时曾指出了LBP特征用于人脸识别问题的优越性，现在可以从实验的角度来考察其效果。这些任务就留给有兴趣的读者。

测试中涉及的test()函数的实现如下，它位于工程目录下的test.m文件中。

```
function test()
% 测试对于整个测试集的识别率
%
% 输出：accuracy --- 对于测试集合的识别率
display(' ');
display(' ');
display('测试开始...');
nFacesPerPerson = 5;
nPerson = 40;
bTest = 1;
% 读入测试集合
display('读入测试集合...');
[imgRow,imgCol,TestFace,testLabel] = ReadFaces(nFacesPerPerson,
nPerson, bTest);
display('..............................');
% 读入相关训练结果
display('载入训练参数...');
load('Mat/PCA.mat');
load('Mat/scaling.mat');
load('Mat/trainData.mat');
load('Mat/multiSVMTrain.mat');
display('..............................');
% PCA降维
display('PCA降维处理...');
[m n] = size(TestFace);
```

```
TestFace = (TestFace-repmat(meanVec, m, 1))*V; % 经过pca变换降维
TestFace = scaling(TestFace,1,A0,B0);
display('..............................');
% 多类 SVM 分类
display('测试集识别中...');
classes = multiSVMClassify(TestFace);
display('..............................');
% 计算识别率
nError = sum(classes ～= testLabel);
accuracy = 1 - nError/length(testLabel);
display(['对于测试集200个人脸样本的识别率为',
num2str(accuracy*100), '%']);
```

15.5 SVM在线资源

作为本章的最后一节，下面介绍两个非常优秀的SVM在线资源供读者学习和研究，在今后的工程实践中也可以十分方便地使用它们。

15.5.1 Matlab的SVM工具箱

互联网上可以搜索到很多优秀 Matlab 的 SVM 工具箱，笔者为您推荐一个由 Steve Gunn (srg@ecs.soton.ac.uk)编写的SVM工具箱，它实现的功能相对集中（支持分类和回归），代码简洁明了，给出了之前介绍的C-SVM的标准实现，二次规划由一个库qp.dll完成，非常适合于初学者理解和掌握，同时它还支持多种核函数。通过对错误代价系数 C 进行设置您还可在最大分类间隔与最小化训练错误代价之间做出权衡。

注意

通过Matlab编写的SVM程序训练一个大样本集合可能导致函数运行缓慢并需要大量的内存。如果在此过程中提示内存不足或者优化过程需要太多的时间，请尝试分割为小一些的样本集合并使用交叉验证的方法来测试分类器性能。

15.5.2 LIBSVM的简介

LIBSVM是台湾大学林智仁(Chih-Jen Lin)博士等开发设计的一个操作简单、易于使用、快速有效的通用 SVM 软件包，可以解决多种 SVM 的相关分类，当然其中也包括我们之前讨论的C-SVM。它提供了线性、多项式、径向基和 sigmoid 4种常用的核函数供选择，采用基于一对一的投票策略解决多类问题，此外还提供了通过交叉验证自动选择最佳参数的实用功能（15.4.5小节）以及对不平衡样本加权和多类问题的概率估计等。

LIBSVM 是一个开源的软件包，需要者都可以免费的从作者的个人主页 http://www.csie. ntu.edu.tw/～cjlin/ 处获得。它不仅提供了LIBSVM 的C++语言的算法源代码，还提供了Python、Java、R、Matlab、Perl、Ruby、LabVIEW 以及 C#.net 等各种语言的接口，可以方便地在 Windows 或UNIX 平台下使用。此外，软件包中还提供了Windows 平台下的可视化操作工具SVM-toy，在进行模型参数选择时可以绘制出交叉验证精度的等高线图。

LIBSVM是以源代码和可执行文件两种方式给出的。如果是 Windows系列操作系统，可以直接使用软件包提供的程序，也可以进行修改编译；如果是UNIX类系统，则必须自行编译。

LIBSVM 在给出源代码的同时还提供了Windows操作系统下的可执行文件，包括进行支持向量机训练的svmtrain.exe，根据已获得的支持向量机模型对数据集进行预测的svmpredict.exe，以及对训练数据与测试数

据进行规格化操作的svmscale.exe。它们都可以直接在DOS 环境中使用。

下载附带的帮助文件中有LibSVM的详细使用方法并配有使用示例，相应的pdf文档中还给出了使用LibSVM的框架[13,14]，其实这也是利用SVM解决实际问题的一个通用框架。在15.4节我们正是遵循这一框架来解决问题的。

参考文献

[1]（美）冈萨雷斯等著；阮秋琦等译.《数字图像处理》（第二版）.电子工业出版社，2003.3

[2]（美）迪达等著；李宏东等译.《模式分类》.机械工业出版社，2003.9

[3]何斌，马天予等著.《Visual C++数字图像处理》（第二版）.人民邮电出版社，2002.12

[4] 四维科技编著.《Visual C++/Matlab 图像处理与识别实用案例精选》.人民邮电出版社， 2004.9

[5] 王艳平，张铮编著.《Windows程序设计》（第二版）. 人民邮电出版社，2008.2

[6] AT&T Laboratories Cambridge.ORL Face database[DB/OL].http://www.cl.cam.ac.uk/research/dtg/attarchive/faced

[7] A Timo, H Abdenour, P Matti, "Face description with Local Binary Patterns: Application to Face Recognition", IEEE Transactions on Pattern Analysis and MachineIntelligence, Vol28, 2006.12

[8]张铮，赵政等.《Expression Recognition Based on Multi-scale Block Local Gabor Binary Patterns with Dichotomy Dependant Weights》，Springer's LNCS 5553，2009.5

[9] 张铮，赵政等.《基于二维MB-LGBP特征的表情识别》. 计算机应用，2009

[10]王宏漫，欧宗瑛.采用PCA / ICA特征和SVM分类的人脸识别.计算机辅助设计与图形学学报. 2003.15(4):416-420

[11]张铮， 白刚，赵政.《3D Representative Face and

Clustering Based Illumination Estimation for Expression Recognition》，Springer’s LNCS 5553，2009.5

[12]张铮.《基于三维特征脸模型的光照参数估计》. 湖南人民出版社，大视野期刊. 2008.10

[13] Chih-Wei Hsu, Chih-Chung Chang, and Chih-Jen Lin. 《A Practical Guide to Support Vector Classification》.Department of Computer Science National Taiwan University, Taipei 106, Taiwan.http://www.csie.ntu.edu.tw/～cjlin.Last updated: December 20, 2007

[14] Chih-Chung Chang and Chih-Jen Lin.《LIBSVM: a Library for Support Vector Machines》. Last updated: June 14, 2007

[15] 张铮，王艳平，薛桂香.《数字图像处理与机器视觉初步》. 人民邮电出版社，2010年4月

[16] Zheng Zhang, ChaoXu, A Noval Distributed Architecture for Expression Recognition, Advanced Materials Research (ISSN: 1022-6680)

[17] Zheng Zhang, et. al, Facial Expression Recognition Based on MB-LGBP Feature and Multi-Level Classification, Advances in Intelligent and Soft Computing(ISSN: 1867-5662), Springer-Verlag

[18] Zheng Zhang, et. al, Wavelet Decompsition and Adaboost Feature Weighting for Facial Expression Recognition. Proceedings of 2011 International conference on Control, Automation and Systems Engineering (CASE), 2011.7, Singapore

图书在版编目（**CIP**）数据

精通Matlab数字图像处理与识别/张铮等编著.--北京：人民邮电出版社，2013.4

ISBN　978-7-115-30463-6

Ⅰ.①精…　Ⅱ.①张…　Ⅲ.①Matlab软件—应用—数字图象处理　Ⅳ.①TN911.73

中国版本图书馆CIP数据核字（2012）第307788号

内容提要

本书将理论知识、科学研究和工程实践有机结合起来，内容涉及数字图像处理和识别技术的方方面面，包括图像的点运算、几何变换、空域和频域滤波、小波变换、图像复原、形态学处理、图像分割以及图像特征提取的相关内容；同时，对于机器视觉进行了前导性的探究，重点介绍了两种目前在工程技术领域非常流行的分类技术——人工神经网络（ANN）和支持向量机（SVM），并讲解了人脸识别的热点技术。

全书结构紧凑，内容深入浅出，讲解图文并茂，适合计算机、通信和自动化等相关专业的本科生、研究生，以及工作在图像处理和识别领域一线的广大工程技术人员参考使用。

精通**Matlab**数字图像处理与识别

◆编著　张铮　倪红霞　苑春苗　杨立红

责任编辑　张涛

◆人民邮电出版社出版发行　　北京市崇文区夕照寺街14号

邮编　100061　　电子邮件　315@ptpress.com.cn

网址　http://www.ptpress.com.cn

大厂聚鑫印刷有限责任公司印刷

◆开本：787×1092　1/16

印张：25.75　　彩插：6

字数：711千字　　2013年4月第1版

印数：1-3500册　　2013年4月河北第1次印刷
ISBN　978-7-115-30463-6
定价：65.00元（附光盘）
读者服务热线：（**010**）**67132692**　印装质量热线：（**010**）**67129223**
反盗版热线：（**010**）**67171154**

链接：http://pan.baidu.com/s/1bnzm9iR 密码：nq1l